



**AN ARTIFICIAL IMMUNE SYSTEM-INSPIRED MULTIOBJECTIVE  
EVOLUTIONARY ALGORITHM WITH APPLICATION TO THE DETECTION  
OF DISTRIBUTED COMPUTER NETWORK INTRUSIONS**

THESIS

Charles Richard Haag, Captain, USAF

AFIT/GCS/ENG/07-05

**DEPARTMENT OF THE AIR FORCE  
AIR UNIVERSITY**

***AIR FORCE INSTITUTE OF TECHNOLOGY***

---

**Wright-Patterson Air Force Base, Ohio**

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

AFIT/GCS/ENG/07-05

**AN ARTIFICIAL IMMUNE SYSTEM-INSPIRED MULTIOBJECTIVE  
EVOLUTIONARY ALGORITHM WITH APPLICATION TO THE DETECTION  
OF DISTRIBUTED COMPUTER NETWORK INTRUSIONS**

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science

Charles Richard Haag, B.S.C.S.

Captain, USAF

March 2007

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

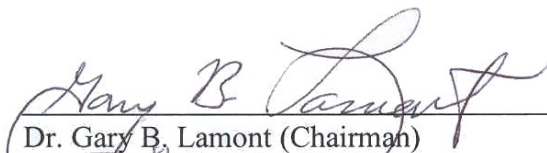
AFIT/GCS/ENG/07-05

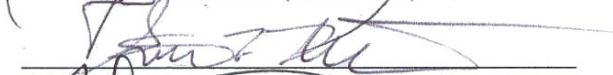
**AN ARTIFICIAL IMMUNE SYSTEM-INSPIRED MULTIOBJECTIVE  
EVOLUTIONARY ALGORITHM WITH APPLICATION TO THE DETECTION  
OF DISTRIBUTED COMPUTER NETWORK INTRUSIONS**

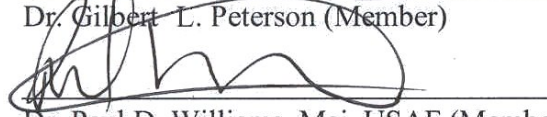
Charles Richard Haag, B.S.C.S.

Captain, USAF

Approved:

  
\_\_\_\_\_  
Dr. Gary B. Lamont (Chairman)

  
\_\_\_\_\_  
Dr. Gilbert L. Peterson (Member)

  
\_\_\_\_\_  
Dr. Paul D. Williams, Maj, USAF (Member)

28 FEB '07  
Date

2 MAR 07  
Date

2 MAR 07  
Date

*Dedicated to my mother of 63 years who unexpectedly passed away near the completion of this research. You never pushed me to be someone you wanted; rather, you trusted and supported every decision I made for myself. I know you'll be at my graduation—just not in the seat next to me.*

## Abstract

Today's predominantly-employed signature-based intrusion detection systems are reactive in nature and storage-limited. Their operation depends upon catching an instance of an intrusion or virus after a potentially successful attack, performing post-mortem analysis on that instance and encoding it into a signature that is stored in its anomaly database. The time required to perform these tasks provides a window of vulnerability to DoD computer systems. Further, because of the current maximum size of an Internet Protocol-based message, the database would have to be able to maintain  $256^{65535}$  possible signature combinations. In order to tighten this response cycle within storage constraints, this thesis presents an *Artificial Immune System*-inspired *Multiobjective Evolutionary Algorithm* intended to measure the vector of tradeoff solutions among detectors with regard to two independent objectives: best classification fitness and optimal hypervolume size. Modeled in the spirit of the human biological immune system and intended to augment DoD network defense systems, our algorithm generates network traffic detectors that are dispersed throughout the network. These detectors promiscuously monitor network traffic for exact and variant abnormal system events based on only the detector's own data structure and the application domain truth set, responding heuristically.

The application domain employed for testing was the MIT-DARPA 1999 intrusion detection data set, composed of 7.2 million packets of notional Air Force Base network traffic. Results show our proof-of-concept algorithm correctly classifies at best 86.48% of the normal and 99.9% of the abnormal events, attributed to a detector affinity threshold typically between 39-44%. Further, four of the 16 intrusion sequences were classified with a 0% false positive rate.

## **Acknowledgments**

My first thanks always to my Lord and Savior Jesus Christ for giving me all I have. Proverbs 16:9 states, “A man’s heart deviseth his way, but the LORD directeth his steps.” It is to my good fortune the LORD decided I should attend AFIT. This knowledge and experience serve a greater purpose I have yet to discover. My sincere thanks and heartfelt appreciation to my thesis advisor, Dr. Gary Lamont for showing me that insight begins with the pedagogical example. I also wish to thank my academic advisor and thesis committee member Dr. Paul Williams, Maj, USAF, for his hours of technical expertise and thesis I first looked at that inspired me to formulate the methodology and mechanics of my core research. Thanks to Dr. Peterson who gave and taught the tools to decipher this research’s data sets, saving me hours of analysis. I also wish to thank Lt. Col. Timothy Halloran, USAF, for his software engineering precepts and provided project skeletons that unwittingly became the foundation of my software design, GUI layout and seamless use of XML in data saving and loading.

My deepest gratitude and appreciation to the United States Air Force for allowing me this highly selective, challenging and unique opportunity to earn my degree from this Air Force’s premier hub for research and development. I’ll always remember the words of former Commandant Brigadier General Mark T. Matthews at my first Commandant’s Call: “You may think coming back to school from the operational world is going to be relaxing and laid-back. To the contrary, this may be the toughest assignment of your entire life.” 21 months, 17 classes and one thesis later, I couldn’t agree more.

Charles R. Haag

## Table of Contents

	Page
Abstract.....	iv
Acknowledgments.....	v
Table of Contents.....	vii
List of Figures.....	xii
List of Tables.....	xvi
List of Algorithms.....	xvii
List of Equations.....	xviii
List of Abbreviations.....	xix
I. Introduction.....	1
1.1 Problem Motivation.....	2
1.2 Research Focus.....	8
1.2.1 Problem Domain Scope.....	8
1.2.2 Approach.....	10
1.3 Research Hypothesis.....	10
1.4 Benchmarks of Validation.....	13
1.5 Thesis Overview.....	14
II. Literature Review.....	15
2.1 The Intrusion Detection System.....	15
2.1.1 IDS Topologies.....	16
2.1.2 Signature and Anomaly Detection Methods.....	20
2.2 The Human Biological Immune System (BIS).....	23
2.2.1 Pattern Recognition, Positive Selection and Negative Selection.....	30
2.2.2 Clonal Selection Theory.....	32



2.3 Artificial Immune Systems (AIS).....	35
2.3.1 Landscape and Ab-Ag Representation .....	38
2.4 Search Algorithms .....	42
2.4.1 Deterministic Search .....	43
2.4.2 Stochastic Search and the Evolutionary Algorithm.....	44
2.5 Multiobjective Evolutionary Algorithms .....	47
2.6 Single and Multiobjective Optimization .....	49
2.7 Pareto Optimality and Nondominance .....	51
2.8 Summary.....	53
III. High-Level Design and Specification.....	54
3.1 Formal Problem Classification .....	54
3.2 Space Complexity and Search Landscape .....	56
3.3 Integrating an MOEA with the Generic AIS Model .....	60
3.4 AIS Application Domain.....	61
3.5 jREMISA: A Continued Work .....	61
3.5.1 REALGO History .....	62
3.5.2 MISA History .....	63
3.6 jREMISA Design.....	64
3.6.1 Data Representation.....	65
3.6.2 Population Initialization and Negative Selection .....	66
3.6.3 Evaluation (Fitness) Functions .....	67
3.6.4 Recombination, Somatic Hypermutation and Affinity Maturation.....	70

3.6.5 Selection Operator .....	71
3.6.6 Detector and Generational Lifecycle.....	73
3.6.7 Calculating the Pareto Front.....	73
3.6.8 Distributed AIS Communication.....	74
3.7 Summary.....	75
IV. Low Level Design and Implementation .....	77
4.1 Hardware and Software Requirements .....	78
4.2 REALGO and MISA C-To-Java Language Translation .....	78
4.2.1 The Model-View-Controller Paradigm .....	79
4.3 Data Signature Design.....	81
4.3.1 Antigen Data Set Encoding .....	82
4.3.2 Antibody Population Generation.....	89
4.4 AIS-Inspired MOEA Pseudocode .....	91
4.4.1 Phase I: Negative Selection .....	94
4.4.2 Phase II: Core MOEA .....	95
4.5 Distributed Communication Model.....	101
4.6 Population Persistence.....	103
4.7 Summary.....	105
V. Experimentation and Analysis .....	106
5.1 Experimental Objectives and Design .....	106
5.1.1 Testing Environment .....	107
5.1.2 Test Functions and Data Sets .....	108
5.2 C-to-Java Migration.....	109

5.3 1999 MIT-DARPA ID Data Set Evaluation.....	116
5.3.1 Negative Selection Results .....	120
5.3.2 Standalone MOEA Results.....	124
5.3.3 Distributed MOEA Results.....	132
5.4 Other MIT-DARPA ID Data Set Evaluation Algorithms .....	134
5.5 Summary.....	136
VI. Conclusions and Future Work .....	138
6.1 Hypothesis Conclusion.....	138
6.2 Conjectures Based on this Research.....	141
6.2.1 Modeling the Innate Immune System.....	141
6.2.2 Protocol-Based Antibody Populations .....	142
6.3 jREMISA: “The Way Ahead” .....	143
6.4 Continued Research Need .....	144
6.4.1 Suitability of the MIT-DARPA ID data sets .....	145
6.4.2 “Cyber Storm”: the next ID data set? .....	147
6.5 Summary.....	147
Bibliography .....	149
Appendix A: ID-Domain Stochastic Search Algorithms.....	A-1
A.1 Simulated Annealing (SA).....	A-1
A.2 Tabu Search (TS).....	A-4
A.3 Genetic Algorithm (GA).....	A-5
A.4 Evolutionary Strategy (ES).....	A-6
A.5 Evolutionary Programming (EP) .....	A-6

A.6 Ant Colony Optimization (ACO) .....	A-7
Appendix B: MIT-DARPA 1999 Week 2 Truth Set Mapping .....	B-1
Appendix C: KDD Cup 99 Data Set .....	C-1
Appendix D: jREMISA User Manual and Source Code .....	D-1
D.1 Quick Start Guide .....	D-1
D.2 User Manual .....	D-3
D.2.1 Compiling and execution.....	D-3
D.2.2 Negative Selection menu (Figure 58).....	D-4
D.2.3 MOEA Menu (Figure 59).....	D-5
D.2.4 “Data Structure [MIT-DARPA 99]” Menu (Figure 61) .....	D-9
D.2.5 “Data Structure [KDD Cup 99]” Menu (Figure 62) .....	D-10
D.2.6 “Packet Ops” Menu (Figure 63) .....	D-11
D.3 jREMISA file hierarchy and UML class diagram .....	D-13
D.4 Special Source Code .....	D-17
D.5 Source Code Availability.....	D-17
Appendix E: Recommended Software Engineering Principles .....	E-1
Appendix F: The Benefits of Open-Source Software .....	F-1
Vita .....	V-1

## List of Figures

	Page
Figure 1: Virus and worm growth trend: Jan 01 – Jun 04 [Symantec04] .....	6
Figure 2: Bot variant growth trend: Jan 03 – Jun 04 [Symantec04] .....	6
Figure 3: Denial-of-Service attack trend, January-June 2006 [Symantec06] .....	7
Figure 4: New vulnerabilities trend, January-June 2006 [Symantec06] .....	7
Figure 5: Example of Ab-Ag complimentary matching [adapted from Timmis04] .....	24
Figure 6: Biological Immune System anatomy (lymphoid organs) .....	27
Figure 7: BIS multi-layer defense structure .....	29
Figure 8: Antigen binding by multiple antibodies .....	31
Figure 9: The Clonal Selection Principle .....	33
Figure 10: Example of shape-space representation of an affinity landscape [adapted from Timmis02] .....	35
Figure 11: Layered AIS framework [Timmis02] .....	37
Figure 12: Recognition region shape-space: a <i>paratope</i> ( $\bullet$ ) recognizes any <i>epitope</i> complement (X) within surrounding volume $V_\epsilon$ .....	39
Figure 13: Hamming distance calculation between two binary molecules of length $L = 8$ .....	41
Figure 14: Pareto front (denoted by bold line) of a bi-objective minimization problem..	52
Figure 15: Two-dimensional search landscape example [adapted from Williams01] .....	58
Figure 16: Negative selection process [HF00] .....	59
Figure 17: REALGO algorithm flowchart [adapted from ELR06] .....	63

Figure 18: Example of transient Ags evaluated against its IP protocol-matching Ab.....	70
Figure 19: Elitist selection operator process .....	72
Figure 20: Mapping of BIS distributed components to a distributed AIS .....	75
Figure 21: Model-View-Controller architecture [Halloran05] .....	80
Figure 22: jREMISA's MIT-DARPA chromosome construction menu .....	83
Figure 23: jREMISA's KDD Cup 99 chromosome construction menu .....	83
Figure 24: IP datagram packet [Stevens94] .....	85
Figure 25: TCP packet [Stevens94] .....	86
Figure 26: UDP packet [Stevens94] .....	87
Figure 27: ICMP packet [Stevens94].....	87
Figure 28: Example encoding of a IP and TCP header to form a TCP DNA chromosome .....	89
Figure 29: Example 240 bit (487-element) TCP Ab chromosome .....	91
Figure 30: jREMISA applied to Timmis' AIS abstract model .....	92
Figure 31: jREMISA algorithm flowchart.....	93
Figure 32: Allele selection process for Cauchy Mutation .....	99
Figure 33: jREMISA distributed communication architecture .....	102
Figure 34: UDP-broadcast payload structure.....	103
Figure 35: Example XML post- <i>negative selection</i> file.....	104
Figure 36: Runtime comparison between REALGO and jREALGO .....	110
Figure 37: Fitness comparison between REALGO and jREALGO: 450 generations....	111
Figure 38: Fitness comparison between REALGO and jREALGO: 5000 generations..	111

Figure 39: Standard deviation comparison between REALGO and jREALGO: 450 generations .....	112
Figure 40: Standard deviation comparison between REALGO and jREALGO: 5000 generations .....	112
Figure 41: Runtime comparison between MISA vs. jMISA.....	113
Figure 42: Statistical runtime comparison between MISA and jMISA .....	114
Figure 43: Plotted MISA, jMISA <i>known Pareto Fronts</i> and MISA's <i>true Pareto Front</i> .....	115
Figure 44: $PF_{\text{known}}$ vs. $PF_{\text{true}}$ point Euclidian-distance differential between MISA and jMISA.....	116
Figure 45: MIT-DARPA “1999 week-two insider” attack data set landscape with LL- labeled attacks .....	118
Figure 46: MIT-DARPA “1999 week-two insider” landscape quantification.....	120
Table 5: Number of generations for each day of the 1999 week-one insider <i>self-only</i> traffic (filtered for TCP, UDP, ICMP only).....	121
Figure 47: <i>Negative selection</i> attrition rate in 1,467,775 generations (Friday) with TCP, UDP and ICMP starting at 4,096 untrained Abs.....	122
Figure 48: Affinity threshold vs. <i>negative selection</i> runtime for TCP, UDP, ICMP = 4096 untrained Abs in 1,467,775 generations (Friday).....	123
Figure 49: Standalone effectiveness against each day of the MIT-DARPA 1999 week-two insider attack data set (39% affinity threshold).....	126
Figure 50: Post-MOEA secondary population <i>true Pareto Fronts</i> .....	129

Figure 51: Post-MOEA <i>attack graph</i> .....	131
Figure 52: jREMISA screenshot of a two-system distributed island model execution ..	132
Figure 53: Standalone vs. distributed effectiveness: Thursday .....	133
Figure 54: Data decomposition: efficiency vs. number of executing jREMISAs .....	133
Figure 55: Warthog vs. jREMISA: false positive rate vs. number of antibodies .....	135
Figure 56: Example of ACO given a preponderance of food at the bottom trail.....	A-8
Figure 57: Ethereal analysis of the 1999 week-two Monday clean insider data set .....	B-1
Figure 58: jREMISA <i>negative selection</i> menu .....	D-5
Figure 59: jREMISA MOEA menu .....	D-8
Figure 60: Example post-MOEA XML output file.....	D-9
Figure 61: JREMISA MIT-DARPA chromosome construction menu.....	D-10
Figure 62: jREMISA KDD Cup 99 chromosome construction menu .....	D-11
Figure 63: jREMISA <i>tcpdump</i> packet operations menu .....	D-12
Figure 64: jREMISA file hierarchy .....	D-13
Figure 65: jREMISA UML class diagram.....	D-16



## List of Tables

	Page
Table 1: History of immunology [DCVZ99a] .....	26
Table 2: Coverage of shape-space ( $C$ ) with required Ab repertoire ( $N$ ) for differing bit string lengths ( $L$ ) and affinity thresholds ( $\varepsilon$ ) with alphabet size $k = 2$ .....	57
Table 3: Antibody signature design [adapted from HWGL02] .....	89
Table 4: MIT-DARPA “1999 week-two insider” attack analysis .....	119
Table 6: Post-negative selection analysis of TCP, UDP, ICMP populations starting at 4096 against the Friday <i>self</i> -only data set of 1,467,775 packets.....	122
Table 7: MOEA run summary: single jREMISA (highest effectiveness in bold text) ...	125
Table 8: MOEA run summary: distributed jREMISA against Thursday data set (highest effectiveness in bold text) .....	125
Table 9: KDD Cup 99 data structure [adapted from KDD99].....	C-3

## List of Algorithms

	Page
Algorithm 1: Bäck's standard Evolutionary Algorithm [Bäck96].....	47
Algorithm 2: Multiobjective Immune System Algorithm (MISA).....	64
Algorithm 3: jREMISA pseudocode.....	94
Algorithm 4: jREMISA negative selection pseudocode.....	94
Algorithm 5: jREMISA fitness function pseudocode.....	96
Algorithm 6: jREMISA selection pseudocode .....	101
Algorithm 7: Basic simulated annealing algorithm [MICHALEWICZ04] .....	A-3
Algorithm 8: Basic tabu search algorithm [MICHALEWICZ04] .....	A-4
Algorithm 9: Genetic Algorithm pseudocode.....	A-6

## List of Equations

	Page
Equation 1: Euclidian distance.....	40
Equation 2: Manhattan distance.....	40
Equation 3: Hamming distance.....	41
Equation 4: Antibody coverage level in Hamming shape space.....	56
Equation 5: Number of Antibodies required for full coverage of Hamming shape Space.....	56
Equation 6: Yao and Liu test function [Yao97].....	110
Equation 7: Kita-proposed function [Kita96].....	114

## **List of Abbreviations**

### Abbreviation

Ab	Antibody
Ag	Antigen
ACM	Association for Computing Machinery
ACO	Ant Colony Optimization
AFIT	Air Force Institute of Technology
AFRL	Air Force Research Laboratory
AI	Artificial Intelligence
AIS	Artificial Immune System
ANN	Artificial Neural Networks
API	Application Programming Interface
BFS	Breadth-First Search
BIS	Biological Immune System
CCNA	Cisco-Certified Network Associate
CDIS	Computer Defense Immune System
DARPA	Defense Advanced Research Projects Agency
DFS	Depth-First Search
DHS	U.S. Department of Homeland Security
DoD	Department of Defense
DoS	Denial-of-Service
GA	Genetic Algorithm

GB	Gigabyte
GUI	Graphical User Interface
EA	Evolutionary Algorithm
EC	Evolutionary Computation
EP	Evolutionary Programming
ES	Evolutionary Strategies
HIDS	Hybrid IDS
HTTP	Hypertext Transfer Protocol
ICARIS	International Conference on Artificial Immune Systems
ICMP	Internet Control Message Protocol
ICSE	International Conference on Software Engineering
ID	Intrusion Detection
IDE	Integrated Development Environment
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IPv4	Internet Protocol, version four
IPS	Intrusion Prevention System
JAR	Java™ Archive File
JAVADOC	Java Documentation
jREMISA	Java-based Retrovirus Multiobjective Immune System Algorithm
JVM	Java™ Virtual Machine

KDD	Knowledge Discovery in Databases
LIMFAC	Limiting Factor
LL	Lincoln Laboratory (at MIT)
MISA	Multiobjective Immune System Algorithm
MIT	Massachusetts Institute of Technology
MOEA	Multiobjective Evolutionary Algorithm
MOP	Multiobjective Optimization Problem
MVC	Model-View-Controller
NIDS	Network-based IDS
NP-Complete	Non-deterministic Turing Machine Polynomial Time-Complete
OS	Operating System
P*	Pareto Optimal Set
PDF	Portable Document File
PF*	Pareto Front
PSO	Particle Swarm Optimization
RAM	Random Access Memory
REALGO	Retrovirus Algorithm
REMISA	Retrovirus Multiobjective Immune System Algorithm
RNA	Ribonucleic acid
SA	Simulated Annealing
SAT	Boolean Satisfiability Problem
SLOC	Source Lines of Code

TCP	Transmission Control Protocol
TISSEC	Transactions on Information and System Security (ACM)
TS	Tabu Search
TSP	Traveling Salesman Problem
UDP	User Datagram Protocol
UML	Unified Modeling Language
URL	Uniform Resource Locator
VPN	Virtual Private Network
XML	eXtensible Markup Language

# **AN ARTIFICIAL IMMUNE SYSTEM-INSPIRED MULTIOBJECTIVE EVOLUTIONARY ALGORITHM WITH APPLICATION TO THE DETECTION OF DISTRIBUTED COMPUTER NETWORK INTRUSIONS**

“The Internet has spawned an entirely new set of criminal activity that targets computer networks themselves. Included in this category are such crimes as hacking, releasing viruses, and shutting down computers by flooding them with unwanted information (so-called "denial of service" attacks). Our vulnerability to – and the damages caused by – this type of crime are astonishingly high.”

*Michael Chertoff, Assistant Attorney General, Criminal Division, U.S. Department of Justice – brief to the Subcommittee on Crime, Committee on the Judiciary, U.S. House of Representatives, June 12, 2001 [Chertoff01]*

## **I. Introduction**

An intrusion detection system (IDS) is a software or hardware device that monitors the events occurring in a computer system or network, analyzing them for patterns of abnormality indicative of a security breach [NIST01]. Signature-based IDSs are naturally reactive and storage-limited. Their operation depends upon experts catching an instance of an intrusion or virus after the potentially successful attack has done its damage, performing post-mortem analysis on that instance, encoding it into an anomaly signature and then storing that signature in its anomaly database. The time required to perform these tasks provides a window of vulnerability to Department of Defense (DoD) automated information systems. Further, because of the current maximum size of an Internet Protocol-based message, the database would have to be able to maintain  $256^{65535}$  possible signature combinations. To best mitigate this vulnerability and limitation, this



thesis presents a proof-of-concept *Artificial Immune System* (AIS)-inspired *Multiobjective Evolutionary Algorithm* (MOEA) intended to measure the vector of tradeoff solution points among detectors with regard to two independent objectives: best classification fitness and optimal hypervolume size.

Modeled in the spirit of the human biological immune system and intended to augment DoD network defense systems, our algorithm generates network traffic detectors that are dispersed throughout the bounded network enclave. These detectors promiscuously monitor network traffic for exact and variant abnormal system events, based only the detector's own data structure and a truth set, and respond heuristically. This research investigates the feasibility of employing such an algorithm in a distributed computing environment to determine if this approach to intrusion detection and classification is more accurate than the single-objective approach.

## **1.1 Problem Motivation**

Signature-based IDSs detect attacks by discovering exact matches between incoming data and a database of known attack string signatures. This reactive nature allows unknown attacks to be successful before the attack signature is defined and stored in the IDS database. In addition, an IDS level of coverage is limited to the resources of the underlying hardware;  $256^{65535}$  possible harmful signatures cannot be stored. Further compounding these constraints, the more storage allotted for signatures, the greater the time required by the algorithm to detect and classify incoming network traffic. These high-level constraints barely skim the surface issues of the intrusion detection (ID) problem domain. Bace and Mell define ID as, “the process of monitoring the events

occurring in a computer system or network and analyzing them for signs of intrusions, defined as attempts to compromise the confidentiality, integrity, availability, or to bypass the security mechanisms of a computer or network” [NIST01]. Reactively performing ID in this manner guarantees two outcomes: high probability of success by every unknown attack and an attack signature database growing beyond the ability of containment.

Therefore, we look toward a proactive algorithm with the potential to effectively classify first-time intrusion encounters without the requirement for an a priori database of intrusion signatures. Developing proactive network defense systems is an open and rarely explored problem. One ID domain algorithm currently being researched is the AIS. Conceived in 1986 [Farmer86], the AIS is inspired by and modeled after the human biological immune system (BIS) for its ability to provide the body the highest degree of protection from invading organisms. Many properties of the BIS are of a growing interest to computer scientists and engineer, particularly those involved in computer security, for the following reasons [DCVZ99]:

1. **UNIQUENESS.** Each individual possesses its own IS, with its own capabilities and vulnerabilities;
2. **FOREIGNER RECOGNITION.** The harmful *non-self* molecules not native to the body are recognized and eliminated by the IS;
3. **ANOMALY DETECTION.** The BIS can detect and react to pathogens never before encountered by the body;
4. **DISTRIBUTED DETECTION.** BIS cells are distributed throughout the body, operating autonomously (no centralized control);

5. IMPERFECT DETECTION (noise tolerance). Exact pattern recognition of pathogens is not required, allowing for variant detection;
6. REINFORCED LEARNING AND MEMORY. Upon disposition of a new pathogen, future encounters are responded to more efficiently and effectively.

Translating these BIS properties into AIS features provides the following benefits for our algorithm:

1. AISs ARE NATURALLY REACTIVE. Signature-based IDSs allow previously unknown nefarious packets to enter and compromise the network because their signature was not in the database. On the other hand, an AIS, which detects abnormal traffic based only on known normal traffic patterns, has the potential to detect, classify and neutralize a newfound intrusion from entering the network (which can be argued as a potential cure to the *Zero-Day Attack*—an attack occurring on the same day or before a defense is created [Porter06]);
2. AN AIS HAS THE ABILITY TO DETECT BOTH EXACT AND VARIANT ANOMALY SIGNATURES. Signature-based IDSs require an exact pattern match, allowing mutated variants of that anomaly into the network. An AIS, conversely, seeks both exact and variant patterns of anomalous traffic, based on a user-defined threshold. When anomalous traffic is confirmed, the detector's data structure slightly changes to include knowledge of this newfound anomaly's structure;
3. AN AIS DOESN'T REQUIRE AN A PRIORI DATABASE OF KNOWN ATTACK SIGNATURES. Rather, an AIS generates a manageable-sized

population of detectors that are initially trained through exposure to known normal traffic and then released to seek network event patterns that do not match such traffic. By having the detector retain the knowledge of newly discovered anomalies dismisses the need for a database of infinite growth size.

However, AISs also have limitations, which should be considered in choosing its application domain. De Castro, classifying the AIS within “nature-inspired computing,” cites nature-inspired computing as [Castro05]:

1. having difficulty in analyzing convergence criteria and optimality of solutions;
2. sometimes not scalable;
3. sometimes inefficient.

Because of today’s exponential proliferation of new and mutated malicious signatures, serially-executed algorithms and deterministic string matching are becoming less efficient, allowing for certain strings to escape into the system. The Symantec Internet Security Threat Report for the first half of 2004 reports alarming growth rates in malicious signatures in that timeframe’s last three years; particularly with regard to viruses, *worms*<sup>1</sup> and *bots*<sup>2</sup> [Symantec04].

---

<sup>1</sup> A computer worm is a self-replicating computer program that sends copies of itself to other computers while executing itself, without user intervention. Unlike viruses, they do not attach to computer files. There are several worm classifications, including instant messaging, file-sharing network and Internet worms [Worm07].

<sup>2</sup> Bots, short for “robots,” are programs that are covertly installed on a user’s machine in order to allow an unauthorized user to control the computer remotely. Bots are used for a wide variety of malicious purposes, such as information theft, stealing application serial numbers, or stealing user passwords. They also facilitate distributed denial-of-service attacks [Symantec04].

Figure 1 shows the exponential growth trend of reported virus and worm signatures while Figure 2 justifies the popularity of signature variants, as equally exponential proliferators.

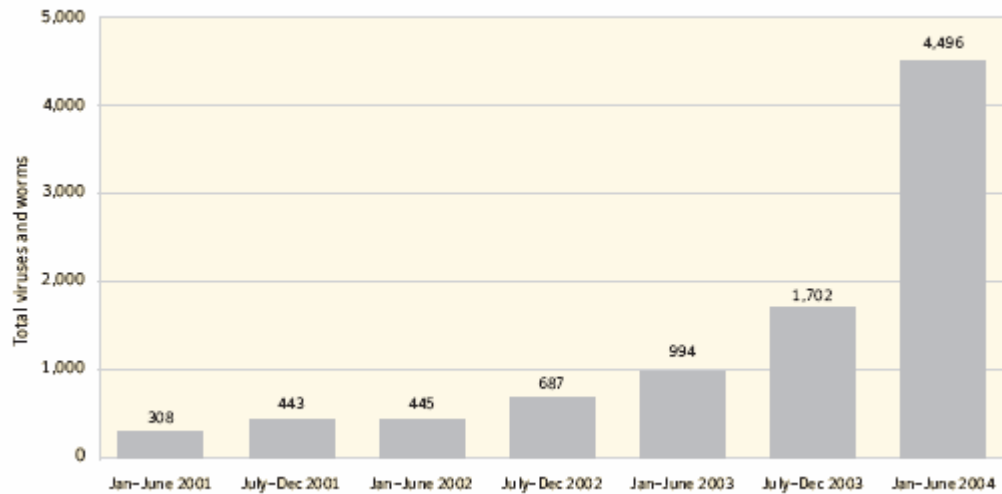


Figure 1: Virus and worm growth trend: Jan 01 – Jun 04 [Symantec04]

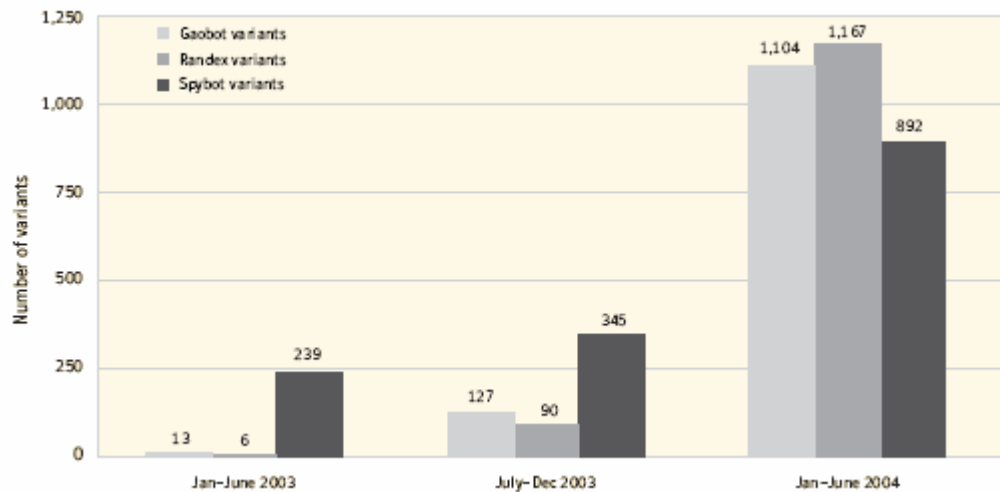


Figure 2: Bot variant growth trend: Jan 03 – Jun 04 [Symantec04]

Symantec's most recent report, which covers the first half of 2006, reports a continued upward trend in malicious activities, i.e., denial-of-service (DoS) attacks

(Figure 3), and new vulnerabilities (Figure 4), maintaining consistency with their 2004 report [Symantec06]. In Figure 4, Symantec comments that the number of vulnerabilities documented in this reporting period is higher than in any other previous six-month period since it began tracking in January of 2002.

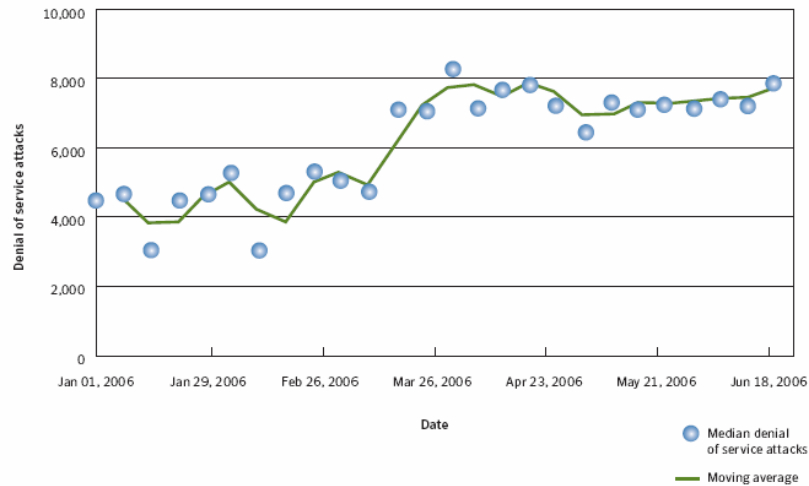


Figure 3: Denial-of-Service attack trend, January-June 2006 [Symantec06]

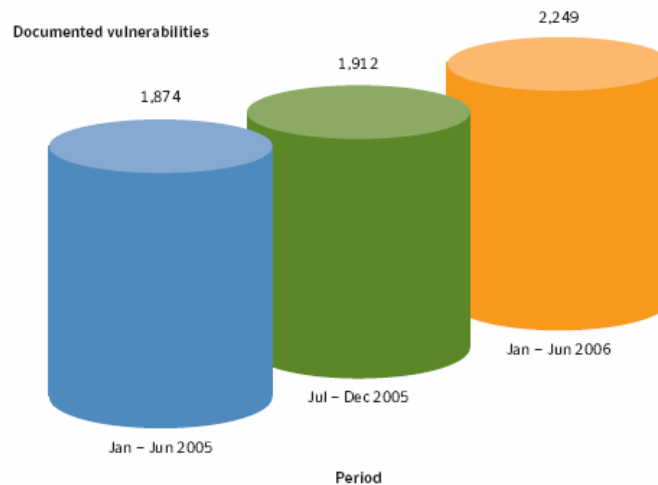


Figure 4: New vulnerabilities trend, January-June 2006 [Symantec06]

## **1.2 Research Focus**

Computer systems are dynamic, with continually changing patterns of behavior, such as management of software applications and users, and continually changing configurations and security policies [HF00]. These and other changes allow intruders to chart methods of gaining improper system access. Traditional computer security mechanisms are mostly static, unable to easily cope with the ever-changing environment. Thus, an adaptive system is needed to track both changes in the environment and the way in which intruders and viruses exploit computer systems. The AIS is the algorithm of choice because the architecture of an IDS is similar to the BIS—a parallel and distributed adaptive system [CC05]. The BIS utilizes volatile memory and is capable of learning and retrieval of information from such memory in recognition and classification tasks. Specifically, it learns to recognize present and past patterns and its global and dynamic behavior impacts many local interactions. These BIS features, in turn, provide robustness, fault tolerance, adaptability and dynamism, which researchers are attracted to emulate.

With regard to the algorithm, the multiobjective context is preferred to the single-objective because reality dictates the ID problem has additional objectives, such as efficiency, effective shaping of the detector for complete search space coverage and measuring individual false detection rates of individual attacks.

### **1.2.1 Problem Domain Scope**

The ID problem domain is too large for only one algorithm's application. It ranges from network-based sniffer systems, responsible for Enterprise-wide coverage, to individual host-based sensors that monitor the activity and usage patterns of a single

system on the network. The algorithm derived from this research addresses ID on the scale of a notional Air Force Base [Mahoney03], defined by the data set, introduced in Section 5.3.

A sub-problem of this domain is the maximization of the inspection coverage of all incoming packets. Ideally, the algorithm should inspect all incoming packets. However, the search space for the current version of the Internet Protocol (IPv4) payload content is a massive  $256^{65535}$  possible strings—too many to search either by deterministic or stochastic means [Williams01, Warthog01]. Hence, our inspection method must be narrowly and heuristically focused to those areas of the search space most profitable to malicious code. For example, there exist only few places within a network traffic packet that facilitate delivery of executable attacks.

One other area of concern is the detection of variants (mutations) of nefarious strings. Slight mutations of existing strings may illicitly enter the system just as easily as a new string could, due to exact signature matching rules. Consequently, a new signature string must be crafted and stored in the database for every possible combinatoric mutation of the known attack string. This introduces storage overhead that could prevent a newly discovered attack signature from being stored. AISs require no more storage space to detect variants than exact pattern matches, depending on the defined matching threshold. However, the risk to variant matching is the possibility of declaring a known event as anomalous. Hence, detectors require constant maintenance in terms of mutating their location and shape to best mitigate fratricide.



### 1.2.2 Approach

This research advances the existing work of two AIS-motivated evolutionary algorithms (EA) applied to the ID problem. These algorithms possess exclusive strengths that we conjecture could be extracted and combined into one algorithm, for synergistic effectiveness. Upon successful translation of each algorithm’s native C programming language to Java, and their subsequent integration into a single Java-based algorithm, this new algorithm is extended to independently execute in a *distributed island model* environment of computers, having the ability to evaluate a data set in a data-decomposition manner [GGKK03]. Whether executed as standalone or distributed, our algorithm is then provided ID data set input for experimental validation. While this algorithm is intended to augment an IDS, the scope of this research allows for only validating such an algorithm’s proof-of-concept and execution.

### 1.3 Research Hypothesis

It is our hypothesis of this research that a “useful” AIS-inspired MOEA can be developed, achieving two independent objectives with regard to detectors:

1. best classification fitness of normal and abnormal traffic;
2. optimal hypervolume size.

The term “useful,” in this hypothesis, is defined by Garrett in his search for how to evaluate an AIS [Garrett05]. Usefulness criteria is based on how distinct and effective a computational method is. If distinctive, it possesses unique symbols or methods that can be transformed to become the same as another method but that its symbols, expressions and processes, as a whole, cannot be made equivalent by another. Effectiveness implies

the accuracy level of obtained solutions in the effort to reach a desired result, or effect, while efficiency stresses minimal computational effort (i.e., time) and resource consumption (i.e., space) by the algorithm [CVL02]. If effective, the AIS must provide a unique means of obtaining a set of solutions, provide better results than other existing methods in a shared benchmark test, or provide more expedient results than other methods in a shared benchmark test.

Through multiobjectivity, a set of globally minimized solutions, rather than a single solution, should provide a greater range of options to network administrators in choosing detectors to employ in future ID applications.

### *Objectives*

Our hypothesis validity is based on a set of quantitatively and qualitatively measurable goals, which is, in turn, based on the outcome of our set of experiments. Given this, our hypothesis goals are:

#### **1. VALIDATE THE MIGRATION OF EXISTING C-BASED AIS**

**ALGORITHMS INTO THEIR JAVA-BASED EQUIVALENTS.** Due to the Java programming language's growing ubiquity [Java04], we decide to continue the work of two existing, C-language AIS algorithms in the Java programming language. Once accomplished, output effectiveness of the Java-based algorithm should mirror that of its C parent. If it does, we have laid the foundation to continue their validated work;

#### **2. ATTAIN THE HIGHEST CORRECT CLASSIFICATION RATE**

**POSSIBLE FOR THIS PROOF-OF-CONCEPT ALGORITHM.** This

objective seeks the highest detection and classification effectiveness rate of detectors. This methodology can generate two types of errors: false-positives (referred to as *Type-I*) or false-negatives (referred to as *Type-II*) errors. False positives are declared conditions or findings that do not exist, such as indicating a normal event as abnormal. Classifying normal as abnormal is synonymous to the BIS side-effect of autoimmunity, where the BIS attacks and kills its own cells—a result of improperly trained detectors (see Section 2.2.1). False negatives are failures to recognize a condition that existed, such as declaring an abnormal event as normal. This results in unrecognized and uninhibited harm in a system. The higher the effectiveness of a detector, the lower this objective’s score. Higher scores resulting from false detections are heuristically determined based on the type and intensity of the detector’s error; hence, we desire the lowest overall score possible for the detector population which translates into its highest effectiveness;

3. **IDENTIFY A KNOWN OPTIMAL DETECTOR HYPERVOLUME.** This objective seeks the optimal hypervolume (i.e., size or *affinity threshold*) of detectors. Research has shown that detector effectiveness is impacted by detector size [McGee07] (and shape [Shapiro05, BDNG06]). Detector size, should not be too high as to react to normal traffic and not too low as to not react to abnormal traffic [Middlemiss06, McGee07]. Hence, in addition to classification fitness, we also desire a detector size deviation value as close to zero as possible;

4. **VALIDATE AIS COOPERATIVE COMMUNICATION WITHIN A DISTRIBUTED ENVIRONMENT.** The components that compose the BIS are

distributed, operate autonomously and cooperatively communicate; hence, it should be modeled as such. As AIS detectors are rewarded for correct classification and detection, the AIS subsequently broadcasts its fittest detectors to all listening AISs, for inclusion consideration into their population. Validation of this objective involves three observable steps:

- a. verification the fittest detectors are broadcast to the subnet;
- b. acknowledgement from listening recipients;
- c. insertion or rejection of that detector into the AISs exclusive population of fittest detectors.

#### **1.4 Benchmarks of Validation**

As Figure 1 and Figure 2 illustrate, nefarious computer traffic is exponentially on the rise, Figure 2 further illuminates the disturbing truth that some of this traffic is meant to travel and execute covertly. Hence, we desire benchmarks and experiments that measure not only overall success levels but also the effectiveness of detecting individual attacks, for the sake of maximizing our level of network security.

Our hypothesis objectives are measured in the following manner:

1. **FIRST OBJECTIVE:** Our first objective is measured by the range of results equality between the existent C algorithms' output and their Java-translated equivalent;
2. **SECOND OBJECTIVE:** As our algorithm is intended to be executed within a dynamic and distributed network, we require a pre-defined data set that simulates this activity. Our chosen real-world data set, containing both normal and

- abnormal traffic, includes a supplementary *truth set* detailing the location and duration of abnormal traffic (see Appendix B). By comparing true and false positives and negatives, plotting the classification of each identified attack and using multiobjective graphing tools, we can measure our success level;
3. **THIRD OBJECTIVE:** This research found no current studies of optimal affinity threshold value. In Chapter 5 experimentation, an optimally known affinity threshold value is empirically derived based on post-execution effectiveness of the data set. Consequently, we desire post-execution detectors that deviate as little as possible from this value. Hence, this objective seeks individual detector affinity threshold deviation, from the empirically-derived value, as close to zero as possible;
  4. **FOURTH OBJECTIVE:** The fourth objective is determined through observation of multiple AISs communicating to each other in the protocol specified in this objective's definition, above.

## **1.5 Thesis Overview**

This chapter provides the problem motivation and meta-level approach behind solving it. Chapter 2 provides the background insight into understanding the basic concepts involved in the scope of this research. Chapters 3 and 4 detail the high-level design methodology and low-level design implementation employed in achieving our hypothesis. Chapter 5 presents the experimental analysis of our constructed algorithm, concluded by the Chapter 6 summary of research impact and future direction of this work.

## **II. Literature Review**

The first step in validating our hypothesis is having an understanding of ID: the history behind it, the lessons learned of past endeavors in solving its problem domain and considering which avenues of current research appear most fruitful in pursuing. This chapter introduces the history and fundamental concepts of anomaly detection, search and evolutionary algorithms, the BIS and solving multiobjective problems to help the reader understand the background behind developing an AIS-inspired MOEA. Section 2.1 introduces the impetus of our algorithm design—the intrusion detection system, with its strengths and weaknesses. Section 2.2 reviews the structure and execution of the BIS that an AIS attempts to computationally model. Section 2.3 reviews the history and advances of the AIS, to date. Section 2.4 discusses the search algorithms applied to anomaly detection. Section 2.5 explains the need for a multiobjective search algorithm over a single-objective. Section 2.6 contrasts the single-objective from multiobjective optimization, with Section 2.7 explaining how to measure multiobjective results.

### **2.1 The Intrusion Detection System**

IDSs are designed to monitor activities on a network and recognize anomalies that may characterize misuse or malicious activity in the form of exact pattern matching and statistical analysis [Chen04]. This recognition idea can be traced back to an early host-based IDS prototype called the Intrusion Detection Expert System (IDES) by James Anderson [Anderson80] and sponsored by the U.S. Navy in the mid-1980s [Navy80]. An IDS consists of three components: monitor, analyzer and responder. Data is collected by

monitoring activities in the host or network. When a suspect event meets a user-defined threshold, a response is triggered. ID approaches can be classified according to monitoring location as host-based, network-based, or hybrid. IDS are further classified by their data analysis approach as being either signature- or anomaly-based detection systems.

### **2.1.1 IDS Topologies**

Before the advent of internal corporate networks' connection to the public Internet, the first-generation IDS was host-based; meaning that an IDS was attached to and monitored a single computer. Subsequently, network-based IDS, executed from a computer connected to a switch or router, are responsible for the monitoring of all passing network traffic. The topology chosen, whether host-, network-based or a hybrid thereof, is driven by the network size and partition(s).

#### *Host-based IDS*

Host-based IDSs operate on information collected from within an individual computer system. From this vantage point, the IDS can effectively monitor all system activities, observe the outcome of abnormal activity, and execute real-time measured response. They normally rely on two system information sources: the operating system audit trails and system message logs. These data are taken together, commensurate with the IDSs own data, to provide reports to the system administrator. Advantages of a host-based IDS include:

1. their locality to the system they are installed on enables their ability to detect certain attacks not able to be seen by a network-based IDS;

2. not being hindered by encrypted network traffic as all data is unencrypted before transmission and post-arrival;
3. unaffected by switched networks;
4. their operation on local operating system audit trails helps detect *Trojan Horses*<sup>3</sup> or other software integrity breach-type attacks.

Disadvantages of a host-based IDS include:

1. their local scope of responsibility prevents their monitoring of network activity (e.g., malicious network scans or surveillance);
2. configuration management-prohibitive, as every IDS installed on an individual workstation or server must be individually managed;
3. susceptible to certain DoS and unmonitored internal or external-threat attacks, allowing for unreported disabling of the IDS;
4. the size of the audit trail utilized by the IDS (i.e., the larger the trail, the more data an IDS has to make informed decisions about authorized activities) is proportional to the amount of space required of the individual computer;
5. local operating system resources are required, thus inflicting a performance cost on a monitored system.

---

<sup>3</sup> A *Trojan Horse* portrays itself as something other than what it is at the point of execution. It neither replicates nor copies itself, but causes damage or compromises the security of the computer. The malicious functionality of a *Trojan Horse* may be anything undesirable for a computer user, including data destruction or compromising a system by providing a means for another computer to gain access, thus bypassing normal access controls, <http://www.symantec.com/avcenter/refa.html#t>.



### *Network-based IDS*

In the interest of centralized management and clandestine IDS sensor (or host) placement, the majority of commercial IDS matured to network-based IDSs (NIDS). NIDS detect attacks by analyzing network packet traffic along a network segment or switch, enabling the monitoring and protection of multiple hosts by a single sensor. NIDS consist of a set of single-purpose sensors placed at multiple, distributed points in the network. The sensor's purpose is to monitor and perform analysis of network packet traffic and report attacks back to the central management agent. These sensors may operate in passive *stealth (promiscuous)* mode in order to make it more difficult for the attacker to specifically seek and identify them. Advantages of NIDS include:

1. a strategically placed few sensors can monitor a relatively large network of hosts;
2. deployment of network-based sensors does not interrupt individual host operation or network communication;
3. ability to be made secure against attack and invisible against detection.

Disadvantages of NIDS include:

1. the volume of network traffic is inversely proportional to the percentage amount of traffic the IDS is able to analyze for anomalies. At peak times of network usage, the IDS may become saturated and unable to inspect all packets, potentially missing a virus or intrusion-related attack packet;
2. increasing the efficiency of packet inspection (in an effort to make contact with all packets) forces vendors to use fewer resources, resulting in the detection of fewer attacks (lowered effectiveness);

3. cannot inspect encrypted packets. This problem may require the ability (and computational overhead) of integrating (versus bypassing) encryption inspection capability, as more companies move toward virtual private networks (VPN);
4. cannot conclude whether or not an attack was successful; only whether or not an attack was present. The human network administrator must conclude the level of success;
5. susceptible to instability and crash due to the inability to handle malformed or fragmented packets, accidental or malicious;
6. many advantages of network-based IDSs don't apply to modern-day switches due to the switch's inability to provide universal port monitoring. This limits the IDS to monitoring a single host. And even with such capability, a single port to a host does not provide a mirror to all ports' traffic.

### *Hybrid IDS*

Referred to by some vendors as, “an IDS that fills more than one role” and “host-based intrusion prevention system (IPS),” hybrid IDS (HIDS) combine the functionality of a host-based IDS and NIDS into one package [SF07]. It binds closely to the OS kernel and services, monitoring and intercepting system calls to the kernel or APIs in order to prevent and log attacks [NSS07]. This topology arose in response to modern switches preventing all switch traffic from being visible to a single host and NIDS tendency to drop packets on high-speed networks. Advantages of HIDS include:

1. effective blocking of attacks against an individual host and its application level;
2. ability to work online with encrypted networks [WS07].

Disadvantages of hybrid IDS include:

1. future OS upgrades could cause problems, based on the OS binding [NSS07];
2. required to be deployed to every host [LJ07].

### **2.1.2 Signature and Anomaly Detection Methods**

IDSs make the distinction between malicious and non-malicious packet traffic based on either a signature (signal) or anomaly (noise)-based configuration to properly identify malicious traffic patterns. In signature-based detection the IDS targets a packet known to be anomalous, is the technique most used by commercial systems. In anomaly detection the IDS hunts for patterns, or signature fragments, of known anomalous packets is the subject of this research. While there are strengths and weaknesses to both, most implementations use a hybrid approach where the preponderance of analysis is signature-based.

#### *Signature-based detection*

By definition, *signature* (or *misuse*) *detectors* look for events that exactly match a pre-defined pattern or signature that describe a known attack or intrusion. These pre-defined (known) signatures are maintained in a database that the detector references. Hence, the effectiveness of the detector is limited to the number of signatures stored. Sophisticated improvements to this method of detection involve leveraging a single signature to detect sets of attacks. Advantages of signature-based detection include:

1. exact pattern matching (vs. anomaly-based) has the potential for the fewest number of false positives (alarms) and, thus, provides the most effective diagnosis as to the specific attack or technique;

2. accurately provides system administrators (of any level of experience) an efficient way to track security problems, enabling them to prioritize their specific security measures.

Disadvantages of signature-based detection include:

1. exact pattern matching restricts detection to only those exact signatures. Hence, the signature repository must be continually updated;
2. detecting tightly-defined signatures prevents the detection of even slight variants of that defined signature.

#### *Anomaly-based detection*

Anomaly detectors are paradoxically different from signature-based in that, by definition, analyze known good (or *self*) traffic over a period of time versus being provided a database of known malicious (or *non-self*) signatures in order to effectively profile anomalous activity within a network. These profiles represent the normal behavior of human users, computer hosts, and network traffic. In this way, incoming data is analyzed to determine if its signature or a variant of deviates from a pre-established norm or exceeds a threshold. This is the method of detection this research focuses on improving. Measures and techniques involved in anomaly detection include:

1. **THRESHOLD DETECTION.** Certain quantitative attributes of user or system activity are recorded as counts with a pre-defined threshold of what is considered acceptable (normal) behavior. Counts can record, e.g., number of file accesses, failed login attempts, CPU utilization by a specific process. This can be a fixed or heuristically updated number;

2. STATISTICAL MEASUREMENT. Includes both parametric, where the distribution of the profile attributes meets a particular threshold; or non-parametric, where the distributed of such attributes is heuristically learned, over time;
3. RULE-BASED MEASURES. Similar to non-parametric statistical measures in that observed activity defines acceptable usage patterns but differs in that these patterns are rule-based versus numeric thresholds;
4. OTHER MEASURES. Today's research involves *artificial neural networks*, *genetic algorithms* (GA), and computational BIS models. Our research involving MOEAs and AISs applies to this area.

Unfortunately, as opposed to signature-based pattern matching, anomaly-based can produce a large number of false positives, as human and computer behavior can be unpredictable. However, conversely, pattern-based matching has the ability of detecting unknown variants that may not have otherwise been detected by signature-based IDSs. In addition, because anomaly-based IDSs identify based on threshold, they require human confirmation of their discovery before reacting—a process called *co-stimulation*. Interestingly, anomaly-based detectors can generate heuristic outputs and signatures, based on its discoveries, which can be used by signature-based IDSs to strengthen their effectiveness. This is one impetus for hybridizing IDS implementations. Advantages of anomaly-based detection include the ability to:

1. detect unusual network behavior without specific or exact knowledge of the attack or signature;

2. automate the process to produce information to augment signature-based IDSs, saving human operator time and resources.

Disadvantages of anomaly-based detection include:

1. requires an initial learning curve: initialization within a typically sanitized (all *self*) network to train *self* detectors to discriminate between *self* and *non-self* network traffic;
2. always a greater probability of false detections over the signature-based method;
3. provides only an approximate solution whereas signature-based ensures exact matching.

## 2.2 The Human Biological Immune System (BIS)

The human biological immune system (BIS) is respected as a highly evolved, decentralized, robust system, charged with providing the human body with the highest degree of protection against various invading organisms (e.g., bacteria, viruses and parasites)—both internal and external to the body [Greensmith03]. It combats dysfunction from a body's own cells and tissues, known as *infectious self*, and the action of exogenous (outside-body) infectious microorganisms known as *infectious non-self* [DCVZ99a]. Collectively, these *non-self* invaders are formally referred to as *pathogens*. The *non-self* pathogen identified in computational circles is commonly referred to as the Antigen (Ag), defined as any molecule recognized by the BIS [Timmis02]. Protection against Ags is achieved through the orchestrated execution of many BIS components, with the most prominent being Antibody (Ab) or *self* detectors. The BIS performs its duty

through recognition and removal of the Ag from the body based on a complimentary matching between an Ab and the Ag, as depicted in Figure 5.

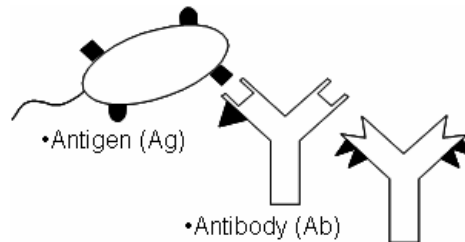


Figure 5: Example of Ab-Ag complimentary matching [adapted from Timmis04]

Without the BIS, death of the body from infection would surely result. Hence, the BIS cells and molecules must maintain constant surveillance for *non-self* organisms. When a pathogen is detected for the first time, it is not only eliminated but its pattern, or signature, is retained in BIS memory so that repeated exposures to the same or variant pathogen are prosecuted more efficiently.

In medicinal history, immunology is a relative new biological science. Its origin is traced back to west-England country doctor Edward Jenner who, in 1796, discovered that human inoculation with the related cow-pox virus built immunity against the deadly scourge of smallpox—a frequently lethal disease. Jenner named this process *vaccination*—the inoculation of healthy individuals with weakened or non-lethal samples of disease-causing agents aiming at educating and consequently strengthening BIS defenses against these specific agents.

Later in the 19th century, following on Jenner's discovery, doctor Robert Koch proved that infectious diseases were caused by *pathogenic organisms*, each producing its

own *infection* or *pathology*. This validation led to the classification into categories of disease-causing organisms called pathogens. Jenner and Koch's discoveries, taken together, formed the basis of the science of immunology. In the 1880s, Louis Pasteur used this knowledge to develop the second vaccine used against *chickenpox*. He called his inoculation an *antirage*. During this same period, Elie Metchnikoff discovered *phagocytosis* and other cellular components, helping define the BIS architecture.

In 1890, Emil von Behring and Shibasaburo Kitasato made the critical discovery that the serum within inoculations contained agents called Abs that bind to infectious agents including Ags and explosively reproduce after exposure to the Ag. At the same time, Paul Ehrlich formulated the *side-chain theory* which conjectured the surfaces of white blood cells, such as B-cells, are covered with several side-chains, or *receptors*. These receptors form chemical links with the complementary links of encountered Ags, allowing for Ab-Ag binding. In the 1950s, McFarlane Burnet proposed the *clonal selection theory* or *clonal selection principle* to help explain the widely disputed circumstances around Ab formation and reproduction [Burnet50], further detailed in Section 2.5.2. An unexplained corollary to Ab reproduction is the possibility of Abs reacting to (destroying) *self-antigens*, which would weaken the BIS. In 1971, Niels K. Jerne argued that the elimination of *self*-reactive cells would constitute a *negative selection* mechanism—a method for eliminating Abs which react to *self*.

In the last few years, most biological immunology is focused on apoptosis, antigen presentation, cytokines, immune regulation, memory, autoimmune diseases,



DNA vaccines, and maturation of the immune response [DCVZ99a]. Table 1 summarizes the research explained in this section.

Aims	Period	Pioneers	Notions
Application	1796-1870	Jenner Koch	Immunization Pathology
	1870-1890	Pasteur Metchinikoff	Immunization Phagocytosis
Description	1890-1910	von Behring & Kitasato Ehrlich	Antibodies Cell receptors
	1910-1930	Bordet Landsteiner	Specificity Haptens
Mechanisms (System)	1930-1950	Breidl & Haurowitz Linus Pauling	Antibody synthesis Antigen template
	1950-1980	Burnet Niels Jerne	Clonal selection Network and Cooperation
Molecular	1980-1990	Susumu Tonegawa	Structure and diversity of receptors

Table 1: History of immunology [DCVZ99a]

The organs and tissues that compose the BIS are distributed throughout the body and known as *lymphoid organs* once they begin production, growth, and maturity of *lymphocytes*—the *leukocytes* that represent the primary BIS operator through recognition and elimination of pathogens. Lymphocytes are composed of two types of cells:

1. *B lymphocyte (B-Cell)*: their main function being the production and secretion of Abs as a response to exogenous proteins (i.e., bacteria and viruses). Each B-cell is programmed to produce a specific receptor-arranged Ab—a property named *specificity*. The successful binding of this B-cell protein to another protein is the signal to other cells that the bound protein must be destroyed;

2. *T lymphocyte (T-Cell)*: named for their maturation within the thymus [Dreher95], they regulate the actions of other cells (i.e., B-cells) and attack host-infected cells.

Lymphocytes become stimulated after an Ag-related interaction, leading to their activation and proliferation. Each lymphoid organ (Figure 6) has a specific defense function. Our area of research is limited to only a few of these organs:

1. *bone marrow*: the soft tissue within the inside part of the longest human bones, responsible for immune cell generation;
2. *lymph nodes*: act as filters, with an internal honeycomb of connected tissue filled with *lymphocytes*, where each node stores immune cells, including B and T cells (and where adaptive immune response occurs);
3. *thymus*: place where few cells migrate to, from the bone marrow, where they reproduce and mature into T cells, capable of producing an immune response.

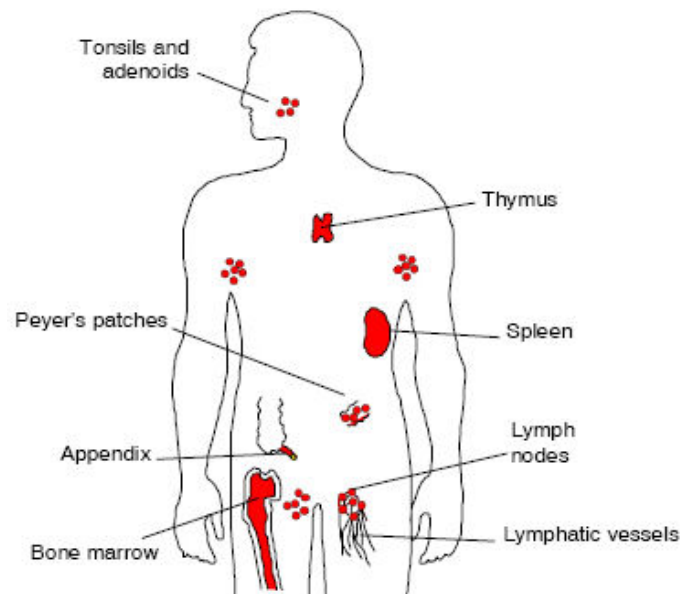


Figure 6: Biological Immune System anatomy (lymphoid organs)

The BIS architecture is a distributed, multi-layered defense system without need of centralized control. The three layers of protection are divided as follows, in Figure 7 [Janeway97; Rensberger96, Hofmeyr00]:

1. *physical barrier (skin)*: our skin is the *firewall* that protects the body from outside invaders, nefarious or otherwise. The respiratory system helps in Ag eradication. Its defenses consist of trapping irritants in nasal hairs and mucus, ejecting them through the act of coughing and sneezing. Overall, it is able to stop some pathogens from entering, however some are able to illicitly enter and confront the second barrier;
2. *biochemical barriers*: bodily fluids, including saliva and tears, contain enzymes that destroy these irritants. Further, stomach acid and temperature kills most microorganisms ingested in food and liquids. These are examples of living conditions that most microorganisms cannot survive in;
3. *innate and adaptive immune system* [Timmis02]: There are two inter-related systems responsible for identifying foreign material within the body. Their functions are described in the next two sections.

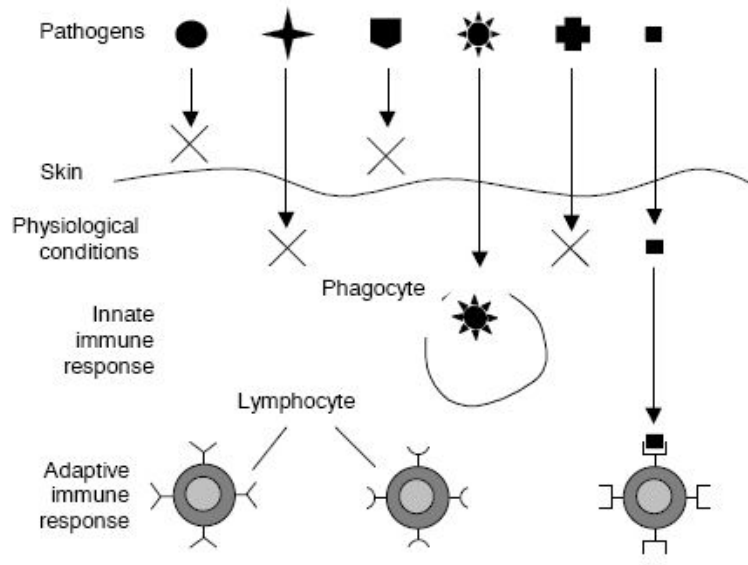


Figure 7: BIS multi-layer defense structure

### *The Innate Immune System*

The innate BIS is the first line of defense against known microorganisms. This means it does not have to first learn about these invaders because such microorganisms are known about at BIS birth. Hence, such internal pathogens are immediately eliminated, ensuring survival at such an immature stage in life. While performing separate functions from the adaptive BIS, the innate BIS is critical in initializing and controlling the adaptive response by controlling/eliminating infection at its level before reaching the adaptive level. Further, the innate BIS plays a role in distinguishing between the *self* and *non-self* and ensures that microorganism structures it recognizes are distinct from the *self*-antigens in order to prevent attacking *self*. The innate BIS can be computationally equated to a basic IDS coupled with a complementary database of known pathogenic (i.e., virus) strings. This half of the IS is not modeled within an AIS (or our algorithm) but is recommended, in Section 6.2.1.

## *The Adaptive Immune System*

All living beings possess a level of resistance to pathogens. The level of resistance is dependent upon the type of organism. If the pathogen is not eliminated by the innate BIS, it then faces the adaptive BIS. The function of the adaptive BIS is two-fold: defeat the pathogen and adapt to its structure so it and any variants may be more efficiently prosecuted in future encounters by lymphocytes—the most important cells of the adaptive BIS. This is computationally equivalent to an IDS without a database of known *non-self* strings scanning for abnormal activity and adding discovered *non-self* strings to a database. Each newly produced lymphocyte (called naïve lymphocytes for their lack of involvement in an immune response) carries a structure of Ag receptors of a single specificity. The arrangement of this specificity is determined during lymphocyte development in the bone marrow and thymus; hence, the chosen structure is combinatoric and specific only to a single Ag. The core of the adaptive immune response is explained by the *clonal selection theory*, introduced in section 2.2.2.

### **2.2.1 Pattern Recognition, Positive Selection and Negative Selection**

Pattern recognition is one of the most important functions of the BIS and is made possible by the Ag-recognizing *surface receptor molecules* of both the B- and T-cells. Immune recognition occurs at the molecular level and is based on the complementarity between the binding region of the Ab receptor, called the *paratope*, and a portion of the Ag called the *epitope* [Timmis02] (Figure 8). While Abs only have a single receptor (specificity) for which to bind to other proteins, Ags have several distinct, complementary *epitopes*, meaning that different Abs can recognize a single Ag.

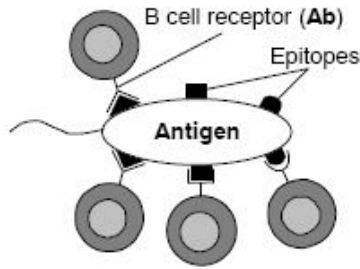


Figure 8: Antigen binding by multiple antibodies

Distinction between what is *self* and *non-self*, which is unknown a priori, is determined by an Ab's immunologic *idiotopes*—epitopes that can be recognized by the Ag binding sites on other Abs. Hence, where Abs bind strongest to Ags of complementary *epitope* arrangement, an Ab can potentially identify another Ab if their receptor arrangement matches. If the BIS cannot perform this distinction, it may be triggered against *self*, causing autoimmune diseases. Conversely, not responding to *non-self* introduces an unacceptable level of tolerance. To solve this *self-non-self* discrimination problem, the BIS performs *positive selection* and *negative selection*.

#### *Positive Selection*

Positive selection ensures the “rescue from cell death” of lymphocytes effective in the Ag recognition process by removing those lymphocytes with ineffective or unproductive receptors [Timmis02]. Hence, Abs producing false detections are eventually removed, allowing the effective Abs the space to survive and clone. This maintains a strong, controlled-size repertoire (population).

#### *Negative Selection*

Negative selection works to prevent lymphocyte fratricide by removing those lymphocytes bearing anti-*self* receptors. Such lymphocytes become *self*-reactive and

attack *self*, potentially resulting in an autoimmune disease. To combat this, the BIS purges the lymphocyte from the repertoire through a lymphocyte-antigen interaction that results in the death (*anergy*) of that lymphocyte [Timmis02]. Put another way, an Ab that attacks *self*, believing it has detected *non-self*, is attacking its own system and must be removed as quickly as possible.

### 2.2.2 Clonal Selection Theory

In 1959, McFarlane Burnet proposed the selective theory that remains scientifically unchallenged as to the most plausible reason for the actions of the adaptive BIS [Burnet50]. The crux of McFarlane's conjecture is that the existence of many cells can produce differing Abs of distinct specificity and similar receptor arrangement as its parent cell. Figure 9 visualizes the *Clonal Selection Theory* (or *Clonal Selection Principle*) during execution. After an Ab binds to an Ag of complementary receptor arrangement (I), accessory (nearby) cells provide a *second signal* (or *co-stimulatory signal*) to allow the Ag to stimulate the Ab. This stimulation causes the Ab to activate and proliferate itself (II) as a *clone*—a cell or set of plasma or *effector cells* (that define the clone size) that are the progeny of the parent cell (III). Further, the B-cell Abs can further differentiate into long-lived *B-memory cells* (which cannot manufacture Abs) (IV). This initial contact between an Ab and Ag is called *primary response*. Based upon this clonal selection theory, the lymphocytes undergo a process similar to Darwin's (1859) *natural selection*.

The mutational and selectional events in the B-cell clonal selection process allow lymphocytes to add to their collection (repertoire) of known *non-self* and increase the Ab-

Ag *affinity*, thus increasing the scope and response time in which known Ags are recognized. Affinity is defined as the strength of binding between the Ab receptors and Ag *epitopes*. Repeated contact with Ags among Abs that matured from the *primary response* allow Abs to prosecute Ags and their possible variants more efficiently and effectively, in future encounters. This is known as the *secondary response* [Timmis02].

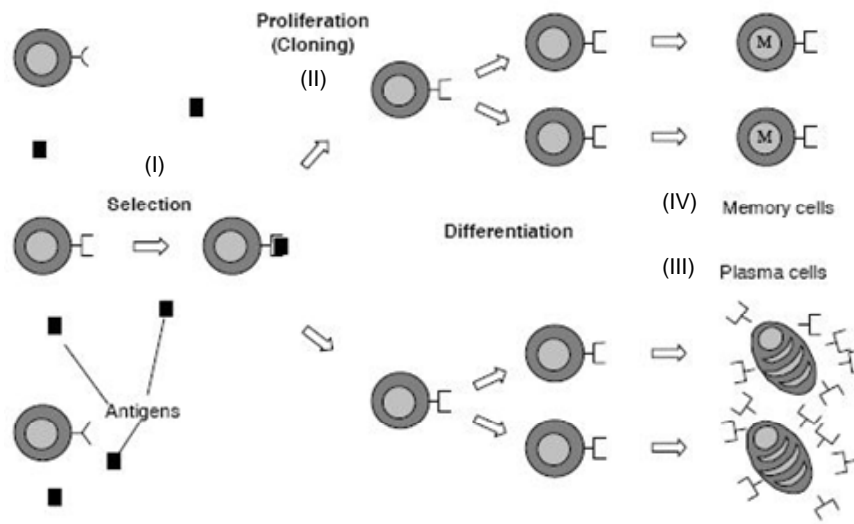


Figure 9: The Clonal Selection Principle

### *Immune Learning and Memory*

Recognition of Ags is not enough; the BIS must have sufficient resources (i.e., storage) to remember future encounters with new Ags, in order to sustain protection of the body. In the BIS, *learning* involves raising the population size and the affinity of those lymphocytes proven effective at recognizing Ags. In handling storage constraints, increases the number of some clones may mean the decrease (forgetting) of others. However, this does not mean the number of lymphocytes has to remain a constant.



### *Hypermutation and Affinity Maturation*

The repertoire of Ag-activated B-cell Abs is improved and diversified via two functions: *hypermutation* and *receptor editing* (or *affinity maturation* of the immune response) of only the high-affinity Abs. Abs involved in the *secondary response* typically have a higher affinity than those of the *primary response*. This *affinity maturation* phenomena is made possible through *somatic hypermutation*. During clonal expansion, random changes are performed on the receptor arrangement of the Ab with the intent of increasing the affinity of the Ab and adding it to the memory pool. However, one must be aware the risk of this random mutation may actually result in a non-functional or *self-reactive* Ab.

George and Gray [GG99] argued for a second diversifying function during *affinity maturation*—Ab receptor editing. They conjectured this would offer the ability to escape *local optima* on an *affinity landscape*. Figure 10 illustrates an example of the purpose of receptor editing [DCVZ99a]. Here, somatic hypermutation with selection aids in discovering the local optima (*exploitation*), while receptor editing introduces diversity to aid in finding the global optimum (*exploration*), which could be a new candidate receptor.

In summary, mutations aid in exploring local regions while receptor editing can prevent premature convergence into unsatisfactory local optima. Hence, successful *affinity maturation* is based upon the complementary roles of these two functions.

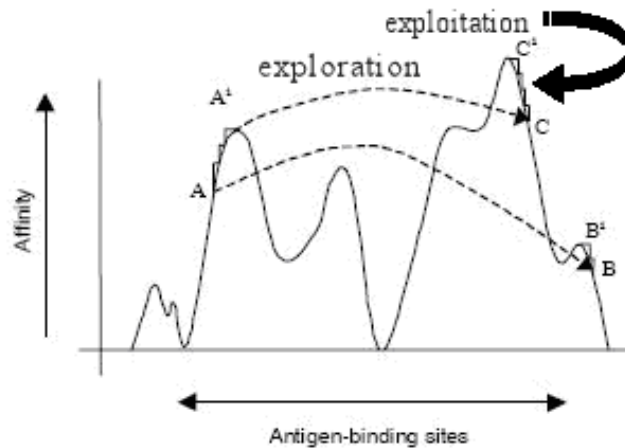


Figure 10: Example of shape-space representation of an affinity landscape [adapted from Timmis02]

### 2.3 Artificial Immune Systems (AIS)

The concept of a computational Artificial Immune System (AIS) was introduced in 1986 by Farmer, Packard and Perelson [Farmer86] who set out to find efficient abstractions of processes found in the human biological immune system. Almost a decade later, AIS practitioners such as Forrest [Forrest95], Dasgupta [Dasgupta99], and Timmis [Timmis02], were motivated to formalize immunological phenomena to develop engineering and computing tools.

AIS is classified as a GA and falls under the field of *Evolutionary Computation (EC)*, defined by Bäck, Fogel, and Michalewicz (who also refer to EC as EA) as, “methods of simulating evolution, most often on a computer. The field encompasses methods that comprise a populated-based approach that relies on random variation and selection” [EC1]. As the AIS is still in its developmental infancy, there currently is no standard definition or experimentally validated problem domain application; a claim

agreed upon at the 2006 (5th annual) International Conference on Artificial Immune Systems (ICARIS). De Castro and Timmis reference three solicited field definitions [Timmis02]:

1. [Starlab]: “*Artificial immune systems are data manipulation, classification, representation and reasoning methodologies which follow a biologically plausible paradigm, that of the human immune system;*”
2. Timmis: “*An artificial immune system is a computational system based upon metaphors of the natural immune system;*”
3. Dasgupta: “*Artificial immune systems are intelligent methodologies inspired by the immune system toward real-world problem solving.*”

These gentlemen sum up these definitions in a single general concept: “*Artificial Immune Systems (AIS) are adaptive systems, inspired by theoretical immunology and observed immune functions, principles and models, which are applied to problem solving.*”

Along with this single concept, De Castro and Timmis attempt to come closer to defining a standard architecture through their abstract AIS model and three required “basic elements” to structure the framework of a biologically inspired algorithm [Timmis02]:

1. REPRESENTATION FOR THE COMPONENTS OF THE SYSTEM:
  - a. Detector: an *Ab* responsible for BIS defense;
  - b. *Ab*: a normal (*self*) network event;
  - c. *Ag*: a abnormal (*non-self*) event, recognizable by the BIS and removed by the *detector*;

2. A set of mechanisms to evaluate the interaction of individuals with their environment and each other (i.e., the environment is shaped by user input stimuli, one or more fitness functions, etc.);
3. Procedures of adaptation that govern the dynamics of the system (e.g., how behavior and chromosomal allele structure vary over time through mutation).

This framework can be thought of as a layered approach (Figure 11). The basis for an AIS begins with the pre-defined problem domain, which governs the method of representation. Once chromosome data structure representation (e.g., bit string, real-valued vector, length, etc.) is decided, one or more affinity measures are used to quantify interactions of the system's elements; e.g., *Hamming distance* measurements applies to bit string representation while *Euclidian distance* is applied to real-valued vectors. The top layer, *immune algorithms*, encompasses those functions that govern the behavior (dynamics) of the system; e.g., method of mutation, selection, evaluation, etc. These layers, integrated into an algorithm, and given a data set of the application domain lead to a potential domain-specific solution.

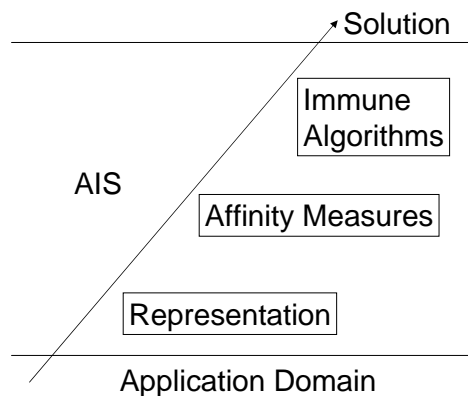


Figure 11: Layered AIS framework [Timmis02]

Initial AIS applications were toward pattern matching, stochastic searchers and sorters of complex problems and data structures [Timmis02]. The realm of computer network security—specifically the intrusion detection problem—is the newest AIS area of research. De Castro and Timmis offer a scope of popular AIS application areas, including but not limited to [Timmis02]:

1. pattern recognition;
2. fault and anomaly detection;
3. data analysis (data mining, classification, etc.);
4. agent-based systems;
5. scheduling;
6. machine-learning;
7. autonomous navigation and control;
8. search and optimization methods;
9. artificial life;
10. security of information systems.

### **2.3.1 Landscape and Ab-Ag Representation**

To computationally model the landscape and actions of the BIS, Perelson and Oster [PO79] introduced the concept of *shape-space* ( $S$ ), as represented in Figure 12. They conjecture a complete repertoire is achieved within the known immune recognition patterns. The concept of shape-space is that the degree of binding between a receptor and the molecule it binds to, a *ligand*, involves short-range interactions based on properties such as hydrogen binding, electrostatic charge, etc. The molecules should be able to

approach and contact an appreciable portion of each other's surface, binding at complementary points. In 1989, Perelson called this receptor arrangement the *generalized shape* of a molecule. Consider that an Ab combining site (*paratope*) can be expressed by  $P$  parameters: length, width and height of any valley or ridge of the combining site; its hydrogen binding degree; etc. Then, a point in a  $P$ -dimensional shape-space specifies the generalized shape of an antigen binding region.

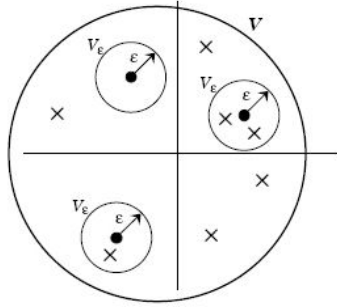


Figure 12: Recognition region shape-space: a *paratope* (•) recognizes any *epitope* complement (X) within surrounding volume  $V_\epsilon$

For example, if a human has a repertoire of size  $N$ , then its shape-space would contain  $N$  points within a finite volume  $V$ . Each paratope can interact with all epitopes within a neighborhood of size parameter  $\epsilon$ , called the *recognition region* of volume  $V_\epsilon$ . The strength (affinity) of an Ab-Ag interaction is measured by its degree of complementarity. The less complementary the interaction, the lower the Ab affinity; albeit the two may still successfully bind. Because each Ab can recognize all Ags in its region and an Ag can have differing *epitopes*, finite Abs can recognize virtually an infinite number of points in  $V_\epsilon$  [DVCZ99].

### *Distance-Measuring Techniques*

The Ab-Ag representation (data structure) aids in determining which distance measure to use in calculating their degree of complementarity (interaction). Mathematically, the generalized shape of a molecule ( $m$ ) can be represented by a set of real-valued coordinates  $m = \langle m_1, m_2, \dots, m_L \rangle$ , which is regarded as a single point in the  $P$ -dimensional real-valued space ( $m \in S^L \subseteq \mathbb{R}^L$ ). The affinity value between the two is related to the “distance measure” between them, as either strings or vectors, using the Euclidian distance or Manhattan distance measure. If the coordinates of an Ab are given by  $\langle \text{Ab}_1, \text{Ab}_2, \dots, \text{Ab}_L \rangle$  and the coordinates of an Ag are similar,  $\langle \text{Ag}_1, \text{Ag}_2, \dots, \text{Ag}_L \rangle$ , then the distance between them is found using Euclidian distance (Equation 1) or Manhattan distance (Equation 2).

$$E = \sqrt{\sum_{i=1}^L (ab_i - ag_i)^2} . \quad (1)$$

$$M = \sqrt{\sum_{i=1}^L |ab_i - ag_i|} . \quad (2)$$

Shape-spaces involving real-valued coordinates and Euclidian distance are called *Euclidian shape-spaces* while shape-spaces involving real-valued coordinates and Manhattan distance are called *Manhattan shape-spaces* [Smith97; S&P88; DeBoer92]. The difference between the two is that Manhattan distance presents an interesting alternative in the domain of parallel (hardware) implementation of these shape-space algorithms [DCVZ99a]. If the molecule’s data structure is represented by a bit string, the

*Hamming shape-space* should be considered for its representation of Abs and Ags as an alphabet of size  $k$  to the power of its length (sequence)  $L$  [Farmer86; Smith97; DBP91; S&C92a,b; Hightower95,96; Perelson96; Detours96]. The Hamming distance (Equation 3) for this type shape-space is defined as

$$H = \sum_{i=1}^L \delta_i \text{ where } \delta_i = \begin{cases} 1 & \text{if } Ab_i \neq Ag_i \\ 0 & \text{otherwise} \end{cases} . \quad (3)$$

When the distance between the two molecules is maximized, a perfect complement is achieved and their affinity is maximal. If not maximal, Ab-Ag interactions in Hamming space must be measured separately from Euclidian and Manhattan space. For Euclidian and Manhattan space, the distance can be normalized (i.e., over parameter  $[0,1]$ ) so that affinity threshold ( $\varepsilon$ ) would also be within that range. On the other hand, if dealing with bit string data structures, then Ab-Ag representation with regard to Hamming distance is simpler. Molecular binding occurs when Ab and Ag bit strings match each other in a complementary fashion. Hence, the Ab-Ag affinity value is the number of complementary bits, determined through application of the XOR operator, as exemplified in Figure 13 [DCVZ99a].

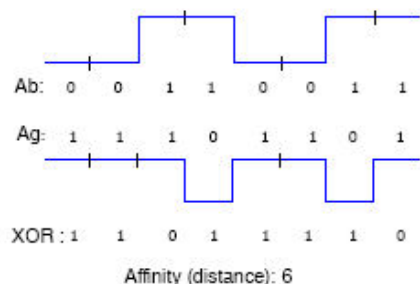


Figure 13: Hamming distance calculation between two binary molecules of length  $L = 8$



An alternative (and complement) to Hamming distance is the *r-contiguous bit rule*, which is considered more biologically appealing [DCVZ99a]. This rule measures affinity through contiguous symbol matching between two sequences. For example, two strings  $s_1$  and  $s_2$  match under the  $r$ -contiguous bits rule if  $s_1$  and  $s_2$  have the same symbols in at least  $r$ -contiguous bit positions (specified by the user). An extensive background and comparison of the various pattern matching functions can be found in [HWGL02].

## 2.4 Search Algorithms

While the AIS is our chosen algorithmic framework, its effectiveness and efficiency are driven by the search algorithm(s) within it. The search algorithm(s) must be carefully selected as each has its own particular strengths, weaknesses and problem domain(s) of application. Many times, in an attempt to utilize an understood algorithm, developers re-shape the problem domain to fit the algorithm. To the contrary, the search algorithm must be selected based on the particulars of the problem domain.

In a given landscape, one seeks either the global optimum value (be it the maximum or minimum) or the set of optimal values. Effective search techniques provide a mechanism for balancing two seemingly conflicting search objectives: exploration and exploitation. Exploration involves maximizing the amount of space searched (diversification) while exploitation exploits a best solution in a localized neighborhood of points (intensification) within the landscape [DPST06]. The purpose of searching is two-fold:

1. at initialization, search for the optimized set ( $PF_{true}$ ) of values to find the fittest detectors in order to most effectively detect anomalies;

2. after initialization (scanning phase), detectors continuously search for these anomalies whose chromosomal composition meet the user-defined matching threshold. Consequently, this effectiveness is based on the search algorithms chosen for the AIS. In this section, the scope of search algorithms is discussed, outlining their strengths, weaknesses, and their domain applicability.

#### **2.4.1 Deterministic Search**

A deterministic algorithm, informally speaking, behaves predictably. Given a specific input, it always returns the same result, passing through the same sequence of states in the state machine. These algorithms are the most popular due to their practicality and efficiency level when executed on real machines. However, as the search space increases in size and dimension, this technique becomes less efficient and feasible.

Strengths of deterministic search include:

1. best at efficiently finding the not-guaranteed optimum value within a neighborhood;
2. guaranteed to terminate within infinite time.

Weaknesses of deterministic search include the inability to:

1. guarantee that final solution is the global optimum (i.e., the algorithm may have found its way into a local optima);
2. escape local optima (without aid of nondeterminism);
3. discretely solve nondeterministic Turing Machine polynomial time Complete (NP-Complete) problems without approximation.

There exist many powerful deterministic algorithms that solve a wide variety of problems, such as greedy search, hill-climbing, branch and bound tree/graph search, depth- (DFS), DFS with backtracking (DFS/BT), breadth-first (BFS) search, and best-first search. However, many multiobjective optimization problems (MOP), defined in Section 2.6, are high-dimensional, discontinuous, multimodal, or NP-Complete [CVL02]. Deterministic methods become ineffective in the face of MOP because they are handicapped by the inability to find the complete set of optimal solutions that composes the Pareto Front (introduced in Section 2.7).

#### **2.4.2 Stochastic Search and the Evolutionary Algorithm**

As the complementary of deterministic, stochastic algorithms are characterized by randomness and unpredictability. Introducing additional decision-makers into the search algorithm, such as probability, system clock time and heuristics, output is no longer the same given the same input. Because many real-world scientific and engineering problems are combinatoric and multiobjective in nature, stochastic search and optimization approaches become preferable to deterministic. However, it should be stressed that one advantage of deterministic algorithms over stochastic, such as DFS/BT, is that deterministic algorithms are guaranteed to terminate because they always can exhaust the entire search space. No stochastic searchers offer this ability [Michalewicz04]. Strengths of stochastic search include:

1. best at approximating the global optimum of a landscape;
2. ability to escape local optima.

Weaknesses of stochastic search include:

1. computationally inefficient when within the locale of a neighborhood;
2. without a termination criterion, it executes for countably infinite time.

### *Evolutionary Algorithms*

While many stochastic search techniques exist, only an EA has the ability to solve multiple problems simultaneously, providing a range of solutions. Other stochastic searchers are constrained to one problem at a time, providing a single solution, and are discussed more in-depth in Appendix A. An EA is a generic term used to indicate any population-based metaheuristic optimization algorithm that uses mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, natural selection and “survival of the fittest” to perform exploration and exploitation [Bäck96]. EAs are initiated with a population of chromosomes—solutions to the search problem we want to solve—in order to find the best solution. The data structure of these chromosomes is defined by the problem domain. In each generation, a set of probabilistic operators are applied to the population of chromosomes, generating new possible solutions. Some of these solutions are then selected to become part of a new, better population. This procedure is repeated until the algorithm has reached a termination criterion defined by the user.

Dr. Thomas Bäck, a fore-thinker in EA theory and practice, mathematically symbolized a standard EA as an 8-tuple [Bäck96]:  $EA \triangleq (I, \Phi, \Omega, \Psi, s, \iota, \mu, \lambda)$

where

- $\mu \triangleq$  number of parent individuals;
- $\lambda \triangleq$  number of offspring individuals;
- $I = A_x \times A_s$  is the space of individuals, where  $A_x, A_s$  denote arbitrary sets;
- $\Phi : I \rightarrow \mathbb{R}$  denotes the fitness function, assigning real values to each individual;
- $\Omega \triangleq \{ \omega_{\Theta_1}, \dots, \omega_{\Theta_z} \mid \omega_{\Theta_i} : I^\lambda \rightarrow I^\lambda \} \cup \{ \omega_{\Theta_0} : I^\mu \rightarrow I^\lambda \}$  is a set of probabilistic genetic operators  $\omega_{\Theta_i}$ , each of which is controlled by specific parameters summarized in the sets  $\Theta_i \subset \mathbb{R}$ ;
- $s_{\Theta_s} : (I^\lambda \cup I^{\mu+\lambda}) \rightarrow I^\mu$  denotes the selection operator, which may change the number of individuals from  $\lambda$  or  $\lambda + \mu$  (depending on the operator's ability to extract good genes from parent to child), where  $\mu, \lambda \in \mathbb{N}$  and  $\mu = \lambda$  is permitted;
- $\iota : I^\mu \rightarrow \{\text{true}, \text{false}\}$  is a termination criterion for the EA, which can be based on a preset number of iterations (generations), an amount of time, or a delta convergence threshold;
- the generation transition function  $\Psi : I^\mu \rightarrow I^\mu$  (“from generation to generation”) describes the complete process of transforming a population P into a subsequent one by applying genetic operators and selection:

$$\Psi = s \circ \omega_{\Theta_{i_1}} \circ \dots \circ \omega_{\Theta_{i_j}} \circ \omega_{\Theta_0}$$

$$\Psi(P) = s_{\Theta_s}(Q \cup \omega_{\Theta_{i_1}}(\dots(\omega_{\Theta_{i_j}}(\omega_{\Theta_0}(P))))))$$

where  $\{i_1, \dots, i_j\} \subseteq \{1, \dots, z\}$ , and  $Q \in \{\emptyset, P\}$ . Bäck defines the high-level algorithmic formulation of EA in Algorithm 1.

---

```

1  procedure BäckEA
2  begin
3     $t := 0$ ;
4    initialize  $P(0) := \{\vec{a}_1(0), \dots, \vec{a}_\mu(0)\} \in I^\mu$ ;
5    evaluate  $P(0) : \{\Phi(\vec{a}_1(0)), \dots, \Phi(\vec{a}_\mu(0))\}$ 
6    while ( $\iota(P(t)) \neq \text{true}$ ) do
7      recombine:  $P'(t) := r_{\Theta_r}(P(t))$ ;
8      mutate:  $P''(t) := m_{\Theta_m}(P'(t))$ ;
9      evaluate:  $P''(t) : \{\Phi(\vec{a}_1''(t)), \dots, \Phi(\vec{a}_\lambda''(t))\}$ ;
10     select:  $P(t+1) := s_{\Theta_s}(P''(t) \cup Q)$ ;
11      $t := t + 1$ ;
12  od
13  end

```

---

Algorithm 1: Bäck's standard Evolutionary Algorithm [Bäck96]

## 2.5 Multiobjective Evolutionary Algorithms

As discussed in section 2.2.2, we desire EAs over other stochastic algorithms for their ability to solve multiple problems simultaneously. However, EAs are typically coded to be constrained to single objective optimization problems. Hence, we consider the *Multiobjective Evolutionary Algorithm* for the following reasons [Lamont06]:

1. **FLEXIBILITY**: can find several trade-off solutions in a single algorithm run instead of a series of separate runs;

2. **CONFIDENCE:** less susceptible to structural forms of solutions—a concern for classical algorithmic techniques;
3. **FEASIBILITY:** find solutions to extremely complex/time consuming and high dimensional real-world applications that have multi-objective goals;
4. **ROBUSTNESS:** use little problem domain knowledge and can generate a good distribution of diverse trade-offs;
5. **IMPLICIT PARALLELISM:** MOEA structures reflect efficient parallel processing.

The definition and generic algorithmic structure of an MOEA is similar to an EA (which is synonymous with EC, per section 2.3) except for minor source code changes to accommodate multiple, independent objectives and their constraints (if any). Per Coello and Cortés, MOEAs share three main similarities [CC05]:

1. **THEY ALL USE PARETO RANKING:** individuals in the population are ranked based on *Pareto Dominance* (i.e., nondominated individuals are scored—or ranked—the highest);
2. **THEY ALL USE SOME FORM OF ELITISM:** this method of selection allows for the retaining of solutions that are globally—not locally—nondominated, with respect to all populations, to include the current one;
3. **THEY ALL EMPLOY DIVERSITY;** i.e., through a mechanism such as mutation.

The primary motivation for using EAs in solving MOPs is their unique ability to deal simultaneously with a set of possible solutions (comprising a population) which allows us

to find several optimally known members of the solution set in a single run of the algorithm, vice separate runs as with traditional algorithms (see Appendix A).

## 2.6 Single and Multiobjective Optimization

As previously discussed, the strength (effectiveness) of Abs is based upon their affinity to Ags. From the affinity landscape perspective, our *objective*, defined as a specified level of desired attainment of a value [CVL02], is the maximal affinity of an Ag binding site. De Castro and Von Zuben define *optimization* as “the task of finding the absolutely best set of admissible conditions to achieve a certain objective, formulated in mathematical terms” [DCVZ99b]. There are three types of optimization within a given data set:

1. discovery of the global maximum;
2. discovery of the global minimum;
3. hybrid: minimization of some values and maximization of others, aggregated into a single objective.

### *Single-Objective Optimization*

Coello, Van Veldhuizen, and Lamont define *global optimization* as, “the process of finding the global minimum<sup>4</sup> within some search space  $S$  [CVL02]. Hence, given a function  $f : \Omega \subseteq S = \mathbb{R}^n \rightarrow \mathbb{R}$ ,  $\Omega \neq \emptyset$ , for  $\vec{x} \in \Omega$  the value  $f^* \triangleq f(\vec{x}^*) > -\infty$  is called a global minimum if and only if  $\forall \vec{x} \in \Omega : f(\vec{x}^*) \leq f(\vec{x})$  where  $\vec{x}^*$  is the global minimum

---

<sup>4</sup> Or maximum, since  $\min \{ \vec{f}(x) \} = -\max \{ -\vec{f}(x) \}$ .



solution(s),  $f$  is the objective function, and the set  $\Omega$  is the feasible region ( $\Omega \subset S$ ). In the context of Ab-Ag affinity matching, we seek the globally optimal (maximal) affinity between an Ab and all Ag binding sites within its reach. In this case, single objective optimization is appropriate. However, reality dictates that an AIS is complex enough to involve more than one objective. Hence, single objective optimization becomes insufficient as we require multiple-objective optimization.

### *The Multiobjective Optimization Problem*

Osyczka defines the *Multiobjective Optimization Problem* as [Osyczka85]: “a vector of decision variables which satisfies constraints and optimizes a vector function whose elements represent the objective functions.” These functions form a mathematical description of performance criteria which are usually in conflict with each other. Hence, the term *optimize* means, “finding such a solution which would give the values of all objective functions acceptable to the decision maker.” In this context, *decision variables* are the numerical quantities for the values chosen for the optimization problem. For example, a vector of  $n$  decisions  $\vec{x}$  is represented by  $\vec{x}=[x_1, x_2, \dots, x_n]$ . *Constraints* mathematically define limitations or restrictions (e.g., resources, physical, time) imposed upon the decision-maker in order to produce an acceptable solution. For example, for a vector  $\vec{x}$  of values to be all positive integers, the function imposed upon it would be written  $g_i(\vec{x}) \geq 0$  for  $i = 1, \dots, n$ . The objective of MOPs is to find good compromises (or “trade-offs”) rather than a single solution, as in global optimization [CVL02]. As more independent objectives are added to a problem, the more complex interpreting the results

becomes. Therefore, we turn to a visualization of the solutions, as conceived in 1896 by Italian economist Vilfredo Pareto.

## 2.7 Pareto Optimality and Nondominance

By definition, MOPs produce multiple solutions which may not be optimal for all objectives [CVL02]. By adjusting one solution for greater optimality, we risk decreasing the desired value of one or more other solutions. Thus, we desire a set or subset(s) of *nondominated solutions* through *Pareto Optimality*. A point  $\vec{x}^* \in \Omega$  is *Pareto Optimal* (with respect to the entire decision space) if there exists no feasible vector  $\vec{x}$  which would decrease some criterion without causing a simultaneous increase in at least one other criterion [CVL02]. Mathematically, for every  $\vec{x}^* \in \Omega$  and  $I = \{1, 2, \dots, k\}$ , either  $\forall_{i \in I} (f_i(\vec{x}) = f_i(\vec{x}^*))$  or there is at least one  $i \in I$  such that  $f_i(\vec{x}) > f_i(\vec{x}^*)$ .

*Pareto dominance* helps to define one vector whose every value is more optimal than another vector. For example, vector  $\vec{a} = (a_1, \dots, a_k)$  is said to dominate vector  $\vec{b} = (b_1, \dots, b_k)$ , denoted as  $\vec{a} \preceq \vec{b}$ , if and only if  $a$  is partially less than  $b$ , i.e.,  $\forall i \in \{1, \dots, k\}, a_i \leq b_i \wedge \exists i \in \{1, \dots, k\} : a_i < b_i$ . In a given MOP,  $\vec{f}(x)$ , the *Pareto Optimal Set* ( $P^*$ ) is defined as a set or subset of nondominated points (i.e., no point dominates another). This is mathematically written as  $P^* \triangleq \{x \in \Omega \mid \neg \exists x' \in \Omega \vec{f}(x') \preceq \vec{f}(x)\}$ .

All feasible solution points are plotted within *decision space*, called *genotype*, and the set of nondominated solutions that rest on the solid boundary region are inside

objective space called *phenotype*, as depicted in the example of a bi-objective minimization problem in Figure 14, is called the *Pareto Front* ( $PF^*$ ) [CVL02].

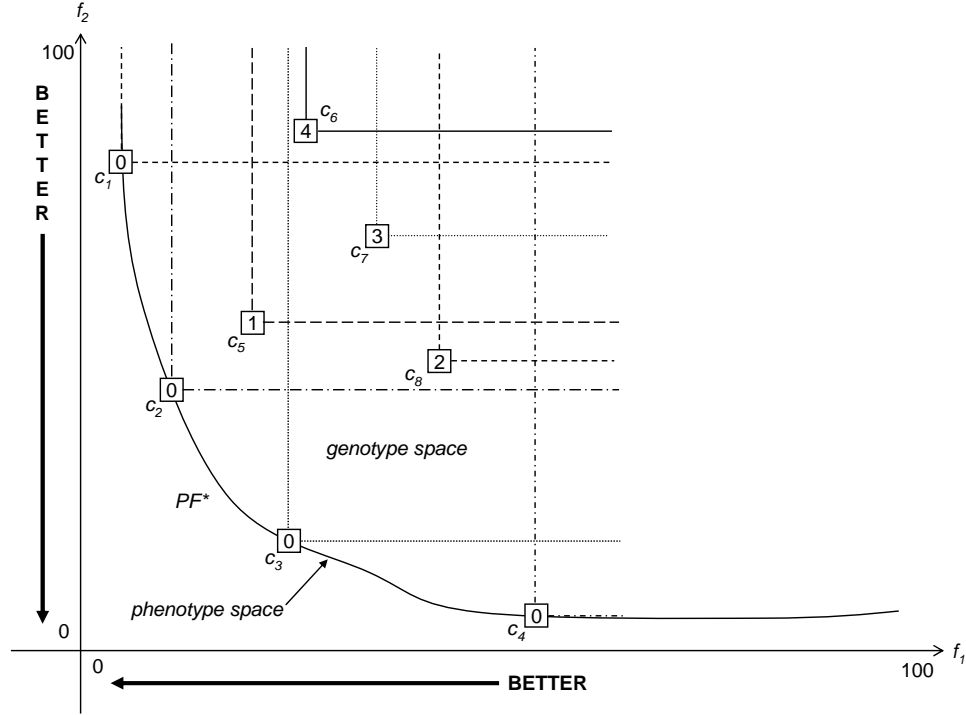


Figure 14: Pareto front (denoted by bold line) of a bi-objective minimization problem

In Figure 14, each solution point has a integer representing the total number of other solution points that dominate it. By definition, all *phenotype*-space points always have a value of zero because they are nondominated. All *genotype*-space points have a value of at least one. A solution point dominates another when it has a value more optimal than another for all objectives. For example,  $c_6$  is dominated by four points:  $c_1$ ,  $c_2$ ,  $c_3$  and  $c_5$ . On the other hand, while  $c_7$  has a lower second-objective ( $f_2$ ) score than  $c_6$ , its first-objective ( $f_1$ ) score is higher; hence, it does not dominate  $c_6$ . Mathematically written,

$$PF^* \triangleq \{ \vec{a} = \vec{f} = (f_1(x), \dots, f_k(x)) \mid x \in P^* \}.$$

Every  $PF^*$  has a *true Pareto Front* ( $PF_{true}$ ) and *known Pareto Front* ( $PF_{known}$ ). A  $PF_{true}$  is the *optimal*  $P^*$  in that no further trade-offs may make the solution set any more desirable (i.e., any increased value of one solution decreases one or more solutions by a greater value).  $PF_{known}$  is the resulting  $PF^*$  upon algorithm termination—it either matches  $PF_{true}$  or is short of it, depending on execution time. Finding  $PF_{true}$  is analogous to executing a stochastic algorithm against an *NP-Complete* problem—the global solution (set) may require infinite time to be discovered. Therefore,  $PF_{true}$  is typically defined by the subjective decision of when to terminate an EA, based on factors such as: setting a fixed number of generations, achieving the pre-determined optimal solution, or lack of further convergence after a number of generations.  $PF_{true}$  is typically used as an effectiveness benchmark against other algorithms' resulting  $PF_{known}$ .

## 2.8 Summary

In summary, this chapter discusses background information relevant to the consideration and construction of an AIS-predicated MOEA with application to the ID and anomaly problem domain. When developing such an MOEA, careful thought must go into the selected problem domain because this drives the choice of search algorithm. Further, the type of optimization must be considered—whether single or multiobjective. To most accurately model a human AIS, a simultaneous problem-solving algorithm should be considered, in a multiobjective context. These ideas form the basis of the high-level design of such an MOEA, in the next chapter.

### III. High-Level Design and Specification

In this chapter, the methodology and meta-level hardware and software architecture design of our AIS-inspired MOEA is presented. To provide perspective, Sections 3.1 and 3.2 discuss the formal classification of our algorithm and the associated space and fitness landscape complexity. Section 3.3 reviews how an MOEA is integrated into the generic AIS model. Section 3.4 formally introduces our algorithm's application domain in order to understand the rationale of our design model. Sections 3.5 provides the impetus behind our algorithm's development. Finally, Section 3.6 explains the algorithm's abstract design and specification.

#### 3.1 Formal Problem Classification

The most common method of identifying computer network intrusions is the matching of incoming network protocol packet headers to “known bad” packet header signature strings [HWGL02], or signature-based detection. This method of string evaluation is analogous to the classic NP-Complete Boolean Satisfiability Problem (SAT) [Michalewicz04]. The SAT is the enumeration (or exhaustion) of a search space of  $n$  variables against a function to determine which variables return true from that function. Because the number of true value combinations can range from zero to one-to-many, the problem degenerates into a worst-case enumeration across the entire search space, leading its classification as a combinatoric NP-Hard problem. By definition of NP-Hard, our problem cannot be solved in polynomial time [DPST06]. Hence, deterministic search methods can take, in worst case, infinite time. Hence, one is forced to consider a

stochastic approximation solution. The amount of data in a packet used to define a signature varies among IDSs, ranging from 49 bits in Hofmeyr's AIS to over 320 bits in Williams' algorithm [Hofmeyr00, Williams01].

This research continues the work of two algorithms formally introduced in Section 3.5: REALGO, which defined a bit string chromosome signature data structure of 30 bits, and MISA, which defined a bit string chromosome of 820 bits. The size of a universe of bit string combinations is dependent upon alphabet cardinality raised to the power of its length. Here, the signature value is either "0" or "1," resulting in an alphabet of size two. This value is raised to the power of the length of the bit string. At a size of 30 bits, there are  $2^{30}$  or approximately 1.07 billion bit string combinations that would need to be generated by a deterministic algorithm in order to completely cover the search space. Exponentially worse, a 820-bit string has  $2^{820}$  or approximately  $6.99 \times 10^{246}$  bit string combinations which must be evaluated.

To determine the time required to evaluate all possible bit string combinations, a simple fragment of Java code was developed to calculate the time required to generate one bit string (chromosome), given a length. Executed on a Windows XP Professional-based operating system (OS) with 1.69 GHz Intel Xeon™ processor and one gigabyte (GB) of random access memory (RAM), the Java Runtime Environment (JRE) version 1.5 required a (340-trial) average of 11.2 microseconds to generate one 30-bit string and 287 microseconds for one 820-bit string. Therefore, one 30-bit string would take approximately 96 seconds to deterministically evaluate while a 820-bit string would take approximately  $7.68 \times 10^{245}$  years. While the latter is clearly unacceptable, the former is

just as unacceptable when one considers the rate at which packets enter the network, which, ideally, should be individually evaluated against all combinations. To compound this problem, only a fragment of all incoming packets represent *non-self* packets, while evaluation on the majority remainder of *self* packets is wasted work.

### 3.2 Space Complexity and Search Landscape

In the Hamming shape-space, the set of all possible Ags is considered as a finite space of points. Ags similar in composition occupy neighborhoods of that space because malicious network activity is typically executed as a sequence of *non-self* packets sent from attacker to victim machine. As exemplified in the last Section, the total number of unique Abs and Ags is given by  $k^L$ , where  $k$  is the size of the alphabet and  $L$  is the bit string length. As depicted in Section 2.3.1, Figure 12, a single Ab can recognize a neighborhood of Ags, based on its *affinity threshold* integer parameter  $\varepsilon$ . For example, if  $\varepsilon = 0$  (i.e., a perfect match is required), then that Ab can recognize only an exact complement Ag. The number of Ags covered by one Ab within a neighborhood (region of stimulation  $\varepsilon$ ) is given by:

$$C = \sum_{i=0}^{\varepsilon} \binom{L}{i} = \sum_{i=0}^{\varepsilon} \frac{L!}{i!(L-i)!}, \quad (4)$$

where  $C$  is Ab coverage [DCVZ99]. Based on Equation 4, an alphabet of size  $k$  and a bit string of length  $L$ , the minimum number of Ab molecules ( $N$ ) needed to fully cover the shape-space is:

$$N = \left\lceil \frac{k^L}{C} \right\rceil, \quad (5)$$

where the value is rounded to the next higher integer. Table 2 gives a perspective on the number of Abs required for full Hamming shape-space coverage, based on varying bit string length  $L$ , affinity threshold  $\varepsilon$ , and alphabet  $k = 2$  (symbols “0” and “1”).

$L$	$2^L$	$\varepsilon = 0$		$\varepsilon = 1$		$\varepsilon = 2$		$\varepsilon = 3$	
		$C$	$N$	$C$	$N$	$C$	$N$	$C$	$N$
2	4	1	4	3	2	4	1	-----	-----
3	8	1	8	4	2	7	2	8	1
4	16	1	16	5	4	11	2	15	2
6	64	1	64	7	10	22	3	42	2
8	256	1	256	9	29	37	7	93	3
16	65536	1	65536	17	3856	137	479	697	95
32	$4.30 \times 10^9$	1	$4.30 \times 10^9$	33	$1.30 \times 10^8$	529	$8.12 \times 10^6$	5489	$7.82 \times 10^5$
64	$1.84 \times 10^{19}$	1	$1.84 \times 10^{19}$	61	$2.84 \times 10^{17}$	2081	$8.86 \times 10^{16}$	43745	$4.22 \times 10^{14}$

Table 2: Coverage of shape-space ( $C$ ) with required Ab repertoire ( $N$ ) for differing bit string lengths ( $L$ ) and affinity thresholds ( $\varepsilon$ ) with alphabet size  $k = 2$

AISs have traditionally focused on the single objective problem because of the argument that the only objective is to effectively classify *non-self*. Single-objective problems are in the form of either one objective or condensing multiple objectives into a single objective, at the cost of effectiveness, in order to find the global optimum. However, reality dictates problems are complex and multidimensional. For example, another independent objective not considered here, to make this a tri-objective optimization problem, is efficiency.

Search landscape dimensionality is driven by data structure composition and the number of objectives. The greater the length (number of dimensions) of the bit string, the more potentially chaotic the search space. Figure 15 depicts a two-dimensional example of a search landscape composed of *self* and *non-self* events [Williams01]. Here, the



landscape is mostly smooth with small clusters of *self* events and fewer, isolated *non-self* events. This is because IDSs have the resources to protect only a limited range of machines and their services. As tighter security policies reduce services or machine coverage, the landscape becomes yet smoother. This observation stems from our data set analysis in Section 5.3.

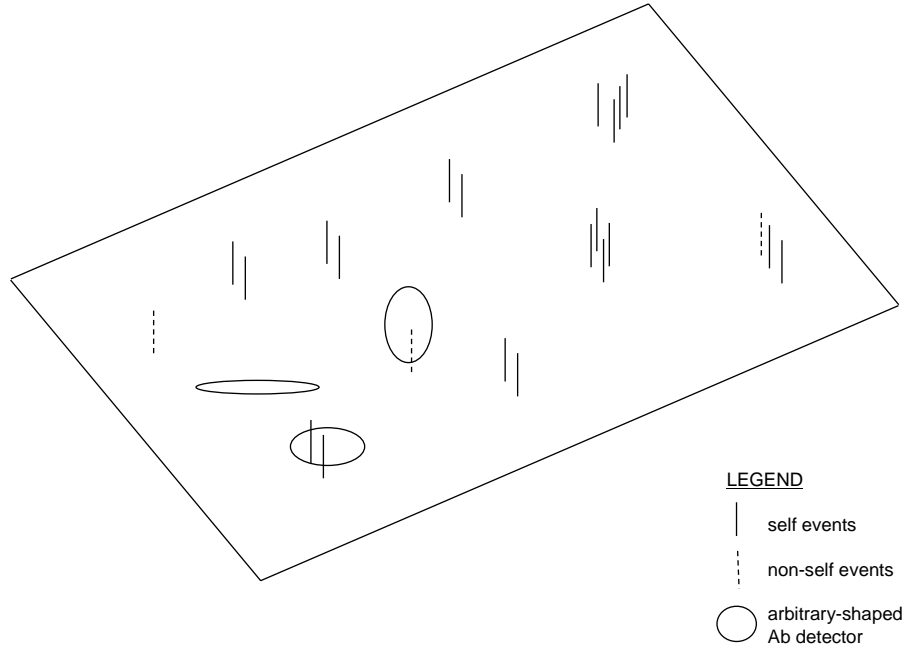


Figure 15: Two-dimensional search landscape example [adapted from Williams01]

Compounding the complexity of this search landscape is the constraint that our Ab detectors must react only to *non-self* events. The location, shape and value of  $\varepsilon$  equate the fitness values of a detector. The larger the value of  $\varepsilon$ , the greater the Ab's neighborhood of detection. However, the tradeoff lies in the greater coverage of *self* events. Hence, an alternative to discarding *self*-matching Abs is to reshape them so they do not cover *self*. Empirical shaping of Abs (e.g., hypersphere, hyperrectangle, hybrid,

etc.) is no meager task, as much research has been dedicated to it; e.g., [Shapiro05]. Figure 15 shows three possible areas of Ab detector coverage: within *self* events, within *non-self* events and in uncovered areas. No coverage is desired in the first area because the detector can declare the *self* as *non-self* (false positive). Further, coverage in the third area is wasteful because no meaningful network traffic resides here. An example would be a detector data structure searching for IP fields or services which security policies have disabled. Hence, all detectors should be shaped to cover areas not inhabited by *self* and non-functional services. *Negative selection*, as discussed in Section 2.2.1, helps to initially shape Ab detectors for ideal coverage by either removing detectors that match *self*, based on an affinity measure as such Hamming distance, or training *self*-matching detectors via mutation until they don't match *self*, as shown in Figure 16. Our algorithm's method of *negative selection*—removal without replacement—is discussed in Section 3.6.2.

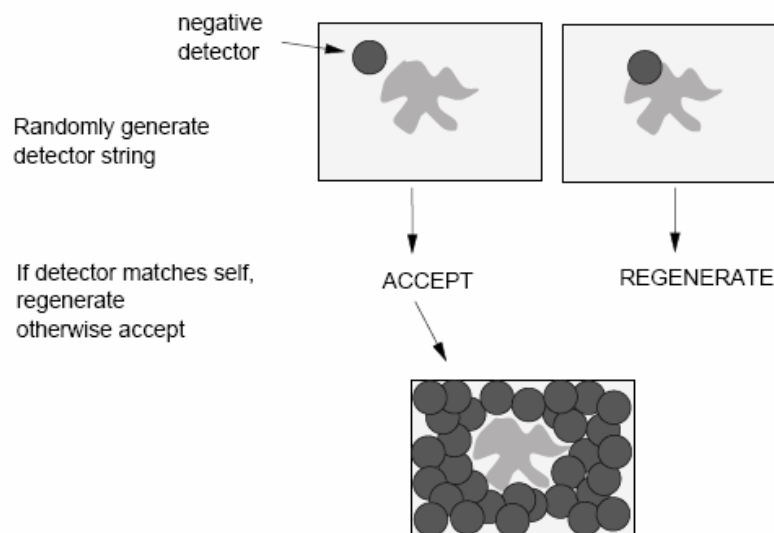


Figure 16: Negative selection process [HF00]

### 3.3 Integrating an MOEA with the Generic AIS Model

As introduced in Section 2.4.2, EAs are defined by their use of evolutionary “survival of the fittest” principles—such as recombination, mutation and selection operators—and a population-based meta-heuristic optimization algorithm to perform exploration and exploitation. Therefore, we consider Bäck’s standard EA construct (see Algorithm 1, Section 2.4.2), expanding the evaluation operator to facilitate two fitness values—one for each objective—to extend this model to an MOEA. Our MOEA’s operators are enhanced by using the ideas of two prior AIS-related algorithms, described in Section 3.5.

As discussed in Section 2.3, only an abstract AIS framework exists, solidified by the problem domain, to guide our AIS construction. AISs are one type of GA, fashioning an EA though efficient abstractions of the human BIS. In defining our AIS framework:

1. our application domain is the ID data set;
2. our representation is a bit string array because of its use by both existing algorithms we build upon;
3. our affinity measure is defined by *Hamming Distance* because it’s the most commonly used method of bit string distance measurement [HWGL02]. We did not choose the *r*-contiguous bit rule because we are pattern-matching the entire context of the packet vs. particular fields;
4. our *immune algorithm* is based on the aforementioned MOEA.

With this algorithm construct and the accompanying ID data set, we possess the required information to develop and test an ID domain solution.

### 3.4 AIS Application Domain

As discussed in Section 2.3, the application domain is the first step in defining the architecture of an AIS algorithm; hence, we now tersely discuss the ID data sets that are input into jREMISA. Because we are analyzing the ID domain on the scale of distributed computers, a data set is required that mimics clean and attack network traffic among many computers. Our chosen data sets are large, binary network traffic files composed of Internet Protocol (IP)-based traffic, with the majority of IP records being Transmission Control Protocol (TCP), User Datagram Protocol (UDP) and Internet Control Message Protocol (ICMP) communication packets. Each protocol, which most commonly facilitates *non-self* traffic [HWGL02], contains packet context headers and content payloads which, when encoded and compared to the Abs, should determine which or not they are *non-self*. In this research, we focus on packet headers only.

### 3.5 jREMISA: A Continued Work

Two AIS-inspired algorithms were found that claimed a level of experimental success over other algorithms with a similar objective: Edge’s Retrovirus Algorithm and Coello and Cortés’ Multiobjective Immune System Algorithm [ELR06, CC05]. Both algorithms were observed to possess exclusive strengths, which, when combined, we conjecture could result in an AIS-inspired MOEA that generates highly effective Abs for

application to an ID data set. For the sake of simplicity and identification, our algorithm is called the Retrovirus-inspired Multiobjective Immune System Algorithm (jREMISA)<sup>5</sup>.

### 3.5.1 REALGO History

The Retrovirus Algorithm (REALGO), a single-objective AIS, was conceived by Edge, Lamont, and Raines, in 2006 [ELR06]. Its intent was to escape local minima when performing complex searches by providing a single memory for each Ab of the last location visited. This practice of preventing premature convergence into local minima was modeled after the BIS' use of reverse transcription ribonucleic acid (RNA). For each generation, the RNA operator copies the Ab into memory. If the next generation results in a lower fitness for that Ab, its RNA copy is restored from the RNA-produced memory in order to continue search in a different direction. Finding the global minimum in this manner is intended to better discover the optimally fit Abs. Therefore, this RNA concept is applied to our fitness function. REALGO's algorithm flowchart, citing the RNA procedure jREMISA utilizes, is shown in Figure 17.

---

<sup>5</sup> Because jREMISA is built upon REALGO and MISA, we recommend the reader review Edge's REALGO paper and Coello's MISA paper [ELR06, CC05].

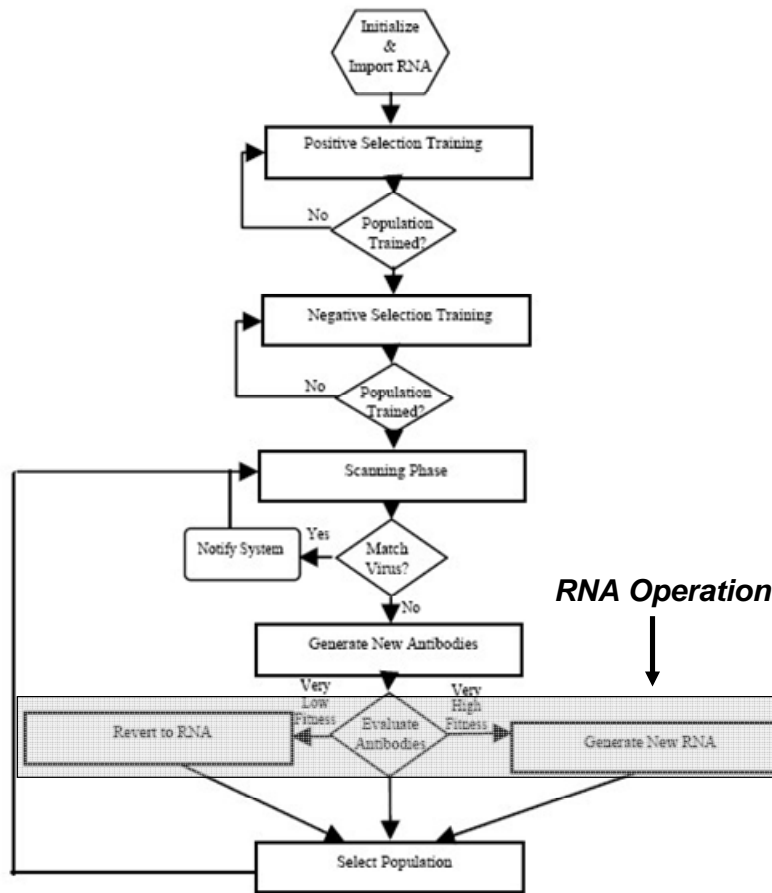


Figure 17: REALGO algorithm flowchart [adapted from ELR06]

### 3.5.2 MISA History

Multiobjective Immune System Algorithm (MISA), a multiobjective AIS based on the *Clonal Selection Principle*, was conceived by Coello and Cortés. They claim this algorithm to be “the first attempt to use an artificial immune system to solve the general multiobjective optimization problem” [CC05]. Their approach uses *Pareto dominance* and *feasibility* to determine which Ab solutions deserve to be cloned. *Nondominated* Abs (solutions) are maintained in a *secondary* (or *external*) population, constituting the *elitist selection* mechanism, which maintains the set of best Abs (solutions) thus far and moves

this  $PF_{\text{known}}$  population toward  $PF_{\text{true}}$  over time. MISA's order of execution is summarized in Algorithm 2.

---

```

1  procedure MISA
2  begin
3    randomly generate initial Population ( $P_i$ )
4    initialize empty secondary Population ( $P_s$ )
5    repeat
6      determine Pareto optimality for all  $Ab \in P_i$ 
7      copy Pareto-nondominated  $Abs \in P_i$  into  $P_s$  /* elitism */
8      determine uniform number of clones for each  $Abs \in P_s$  to expand
         $P_s$  by 600%
9      perform cloning of  $Abs \in P_s$  based on Step 8
10     apply uniform mutation to each cloned  $Abs \in P_s$ 
11     replace lost  $Abs \in P_i$  by copying back best  $Abs \in P_s$ 
12  until (predetermined number of evaluations)
13  end

```

---

Algorithm 2: Multiobjective Immune System Algorithm (MISA)

However, MISA differs from other MOEAs in that it does not use recombination due to the sufficiency of mutation to move its Abs around the search space. Because of MISA's experimental results, our algorithm utilizes their *clonal selection principle* methodology by implementing a secondary population and mutation and selection operators according to their specification.

### 3.6 jREMISA Design

This section provides a high-level, abstract overview of the methodology and many factors that are integrated into jREMISA. Single objective EAs consist of a population of Abs where each has a singular *fitness value*. This *fitness value* allows Abs to be ranked amongst each other at each generation, enabling a *selection* mechanism to

decide which Abs survive to the next generation ( $\iota+1$ ). Being multiobjective, each of our solutions (Abs) contains a set of two values:

1. an integer measure of how effectively they classify between *self* and *non-self*;
2. an integer measure of their affinity threshold (hypervolume) deviation from the starting affinity threshold defined at *negative selection*.

We desire a global minimum because:

1. a higher fitness value means more penalties have been assessed for incorrect classifications;
2. we desire Ab hypervolume to deviate as little as possible from the experimentally derived ideal affinity threshold of 39% (see Section 5.3).

Consequently, multiobjective algorithms don't return a globally optimal solution but rather a set of solutions, allowing analysis of the *Pareto Optimal* solution set's tradeoffs through a *Pareto Front*.

### 3.6.1 Data Representation

Per Section 2.2, the two key actors are the Abs and Ags. Abs are the BIS detectors equally distributed throughout the body, searching for *non-self* Ags. Ags come in two forms: *self* Ags (normal traffic events) and *non-self* Ags (abnormal traffic events). A single Ab or Ag is referred to as a *chromosome* where each dimension is referred to as an *allele*. It is the duty of the Ab to interact with Ags to classify them as either *self* or *non-self*. Hence, Abs are system defenders while incoming information from the outside is considered *pathogenic*, coming into contact with Abs to determine if the percentage of



complementarily over chromosome length meets the affinity threshold for the Ab to declare the Ag as *non-self*.

As introduced in Section 2.3.1, the data structure that compose Abs and Ags can be formed of differing data types (e.g., integer, binary, real-valued numbers, etc.), alphabets and lengths, which determine the dimensionality of the search landscape. AIS algorithm designers typically employ fixed-length binary string representation due to its ease of manipulation by EAs, low computational cost, minimal size, and, most importantly, it most closely models the BIS for its simple “yes-no” complementary matching outcome of Ab-Ag epitope encounters (i.e., affinity-matching Hamming value). Further, this was (conveniently) the data structure employed by both our prior algorithms. Therefore, we encode our generated Abs and incoming data set Ags as bit strings. Bit string length is determined by the type of each incoming packet, further discussed in Section 4.3. Because we chose the signed integer data type for our data structure, we have the freedom of assigning values to Ab alleles other than zero and one. Using Java, our allele values can range between  $-(2^{31})$  and  $2^{31}-1$ . Hence, seven additional attributes are appended to the end of each Ab: name, number of false detections, (true positive + true negative) fitness score, (false positive + false negative) fitness score, affinity threshold deviation, whether or not the Ab has been broadcasted to the subnet and *Pareto dominance* value.

### **3.6.2 Population Initialization and Negative Selection**

The single population of Ab detectors is typically initialized through a pseudorandom-generated binary value for each allele of the Ab array. This was the

method employed by both REALGO and MISA. Our method generates values the same way but differs in population pool size and Ab length. To date, all AIS literature has suggested initiating a single randomly-generated pool, in which all trained Abs are evaluated against each incoming Ag. However, since we have the ability to determine our incoming data packet's protocol (i.e., TCP, UDP or ICMP) before it is evaluated, only Abs of the same protocol are evaluated against the Ag. Hence, we initialize three separate fixed-length populations whose length always matches that of the incoming Ag. We conjecture this increases fitness function accuracy by evaluating Abs and Ags of matching protocol and increases efficiency by limiting *negative selection* evaluation to one subset of the three Ab populations.

*Negative selection* is performed with a user-defined affinity threshold and Hamming distance measure. If the total complimentary bits divided by the length of the bit string meets or exceeds the threshold, it is discarded without replacement. This is preferred over mutation training to guarantee the surviving population doesn't recognize a single *self* event in the clean data set. This function does not employ the data set truth set.

### 3.6.3 Evaluation (Fitness) Functions

As introduced in Section 2.2.1, pattern recognition based on the complementarity of binding regions between an Ab and Ag is the heart of our fitness function. This algorithm has three different *evaluation functions*: *negative selection*, *fitness function* and the *Pareto optimality* test. With our random populations established, the data set stream is opened, reading in one packet at time, in a “sliding window” fashion, as depicted in

Figure 18. Each *tcpdump* packet received is encoded into an Ag and its affinity from each Ab measured using Hamming distance (from Section 2.3.1, Equation 3):

$$H = \sum_{i=1}^L \delta_i \text{ where } \delta_i = \begin{cases} 1 & \text{if } Ab_i \neq Ag_i \\ 0 & \text{otherwise} \end{cases} .$$

### *Negative Selection*

The first *evaluation function*—step three of Bäck’s EA—occurs only once and represents the *negative selection* phase—the removal of all random Abs that match *self*. Here, the data set is sanitized, containing only *self* events, for the purpose of training the random Abs not to react to *self*. As an Ab and Ag are compared, the distance value is divided by the Ab length to determine if the value meets or exceeds the user-defined affinity threshold. If so, the Ab has reacted to (and would summarily attack) *self*; hence, it is discarded. At the completion of *negative selection*, remaining Abs are *feasible solutions* that meet our *constraint* of not matching *self* and are titled, “trained but immature:” trained to discern *self* from *non-self* but immature in lack of contact with *non-self*. These trained populations are now ready to interact with an ID data set containing labeled attacks.

### *Fitness Function*

The second *evaluation function*, which occurs within each generational loop, is the *fitness function* responsible for calculating fitness values of each Ab against a data set with labeled attacks. Per Section 2.6, optimization involves finding the global maximum or minimum value within a landscape. We desire the global minimum for our independent objectives. The first objective measures the sum value of correctly classified

*self* and *non-self*, in which we desire a minimal value. The second objective measures Ab hypervolume deviation, in which we desire a minimal value, as well. As this is a proof-of-concept algorithm, a *truth set* of extracted attack packet numbers guides the *fitness function* to determine whether the Ab was correct in its classification (see Appendix B). Every Ab suffers both a 50% chance of Cauchy mutation and a penalty value in its fitness (first) objective axis at every generation based on how inaccurate their Ag classification was. Hence, the fittest Abs have the lowest objective fitness scores. Each correct Ab declaration rewards it with a 1% increase in affinity threshold (hypervolume) and a copy of its chromosome within its RNA space. Conversely, a false detection shrinks that Ab by the same affinity rewarded, assesses a “false detection” point, and its chromosome is reverted to its stored RNA, returning it to its last search space position. If the number of false detections equals the user-defined Ab lifespan, the Ab is removed from the population.

#### *Pareto Optimality Test*

The third *evaluation function* measures the *Pareto optimality* of each Ab. As introduced in Section 2.7, nondominated points are those desired solutions that lie within *phenotype* or *objective space*. Every Ab is compared to every other Ab with regard to dominance cardinality. Upon completion of scoring, *Quicksort* sorts the array in dominance-ascending order to minimize the search time for the *selection* operator when copying qualifying fittest Abs from the primary to the secondary population.

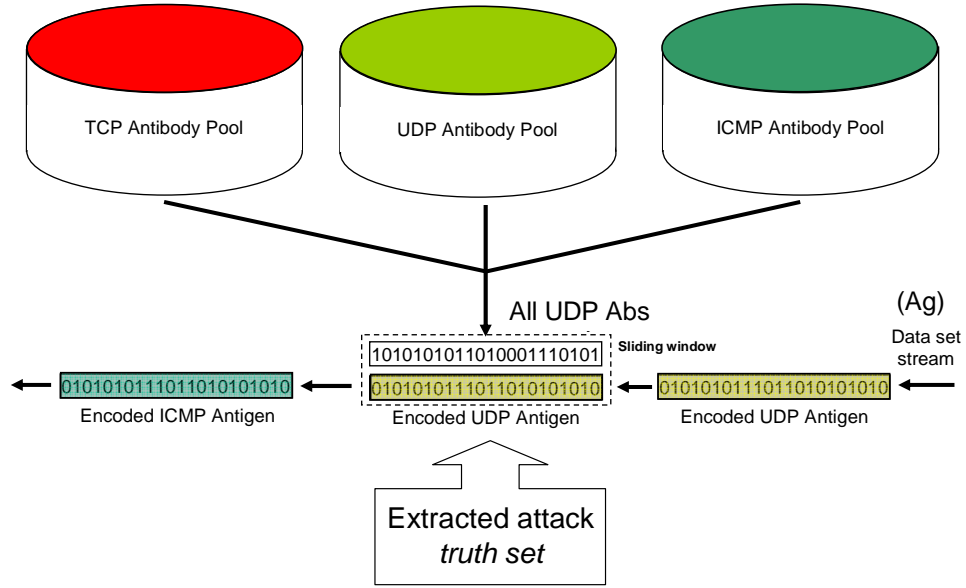


Figure 18: Example of transient Ags evaluated against its IP protocol-matching Ab

### 3.6.4 Recombination, Somatic Hypermutation and Affinity Maturation

REALGO and MISA did not utilize recombination (crossover) due to the sufficiency of mutation to move Abs around the search space and assimilate encountered *non-self* data structures. Therefore, jREMISA does not employ crossover. Two types of mutation are employed at two different areas of jREMISA:

1. **CAUCHY MUTATION.** Cauchy (distribution) mutation, used in REALGO, is the division of a Gaussian distribution-generated random number by itself. This mutation has been shown to have the ability to make long jumps to escape local minima as compared to a Gaussian distribution [Yao97]. Hence, we use this mutation method on false-detecting Abs in the *fitness function*. While REALGO sets a 50% chance for each allele (bit) in the array to be mutated, we heuristically

choose which alleles receive Cauchy mutation, based on whether it was a false positive or negative, described further at the end of Section 4.4.2.

2. **UNIFORM MUTATION.** This method, recommended by MISA, is employed in our *selection operator* upon all cloned Abs within the secondary population. The nondominated clones are mutated in  $N$  random positions, where  $N$  is the number of objective variables. Dominated solutions are randomly mutated in ( $N$  plus the number of Abs dominated by) positions.

*Affinity maturation* is the process of enlarging the Ab volume with the intent of covering as much *non-self* space as possible, without impinging on *self* space. Our Ab affinity deviation value is adjusted based on the truth of the Ab declaration, within the *fitness function*. For every correct classification, the Ab matures (increases its affinity threshold, or hypervolume) by 1%; otherwise, it decreases by 1%.

### 3.6.5 Selection Operator

Following MISA design, our *selection operator* involves a secondary or external population that manages nondominated solutions. It employs *elitism*, copying the top 5% of nondominated solutions from the primary into the second population. If there are not enough nondominated solutions to compose 5%, then the least nondominated solutions fill that gap. Next, the *clonal selection principle* (see Section 2.2.2) selects the Abs most effective at Ag detection for cloning and subsequent mutation. Coello and Cortés performed a series of sub-experiments to determine the most effective cloning rate for each fittest Ab, concluding the Abs should be uniformly cloned until the secondary population increases to 600% its size [CC05]. This cloning operation is followed by

*somatic hypermutation* (mutation) using a MISA-suggested *uniform distribution*. Nondominated solutions are mutated in  $N$  random positions, where  $N$  is the number of objectives. Dominated solutions are mutated in ( $N$  plus the number of Abs dominated by) positions. The fittest secondary Abs are then copied back into the primary until the primary population returns to original size (if any primary population Abs were removed due to their meeting the false detection threshold), defining our evolution as a mix of both parents and children ( $\mu + \lambda$ ). Finally, all dominated solutions within the secondary population are removed, restoring it to a nondominated pool. The secondary population cannot exceed the size of the primary population. This process is depicted in Figure 19.

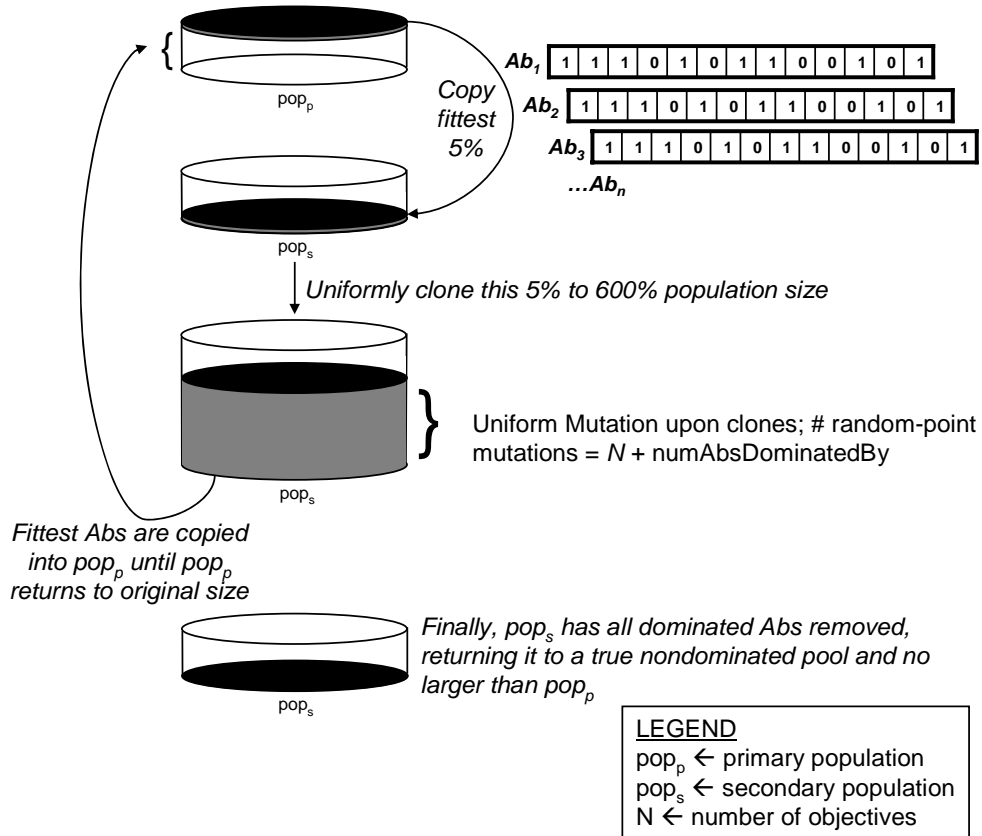


Figure 19: Elitist selection operator process

### 3.6.6 Detector and Generational Lifecycle

Ab detectors do not live indefinitely unless they maintain nondominance. Those detectors that meet the threshold for number of false detections are declared ineffective and are discarded so more effective detectors may inhabit the search landscape. The number of generations of an EA depends on the application domain. Algorithm termination is typically based on:

1. a fixed number of generations  $t$ ;
2. a measure of convergence over a set number of generations  $t$ ;
3. a lack of a significant increase in fitness over  $t$  generations;
4. the desire to allow indefinite execution ( $t \leftarrow \infty$ ), as in a live environment.

jREMISA considers the passing of each data set's Ag (packet) to be one generation vs. the size of the data set. Hence, our number of generations is the number of packets (events) of the entire data set.

### 3.6.7 Calculating the Pareto Front

MISA defines its *true Pareto Front* as the secondary population, at algorithm termination [CC05]. The individuals within are all nondominated not only with respect to each other but also with respect to all previous Abs attempting to enter this population. Per our concept of three initial populations, jREMISA, in turn, has three secondary population *Pareto Fronts*: one for TCP, one for UDP and one for ICMP. Any dominated points within the secondary population at algorithm termination are indicative of not enough nondominated points in that population. The values of these sets can then be mapped into a two-dimensional *Pareto Front*.



### 3.6.8 Distributed AIS Communication

The BIS is, among other traits, a parallel and distributed system. Abs and other BIS cells roam throughout the system, operating autonomously, yet communicating to each other (e.g., an Ab communicates to nearby Abs when it has been stimulated by an Ag). The Air Force Institute of Technology (AFIT) developed the Computer Defense Immune System (CDIS) in an effort to combat the computer virus problem in a proactive manner [HWGL02, Marmelstein99]. CDIS is a multi-agent, hierarchical, distributed computational immune system modeled after BIS archetypes. In the distributed context, CDIS addressed the need to disburse Ags and their workloads among the nodes in networks. Lippmann, in his recommendations to improve existing IDS, recommends approaches to detecting new attacks—specifically anomaly detection—should be extended to multiple hosts [Lippmann00].

Our algorithm furthers these ideas through facilitating two different types of messages to other jREMISAs running on the same subnet: newly discovered nondominated Abs and user-typed instant messaging. A single AIS can monitor only a single host or segment of network traffic, covering only a small portion of the search landscape. Employing multiple AISs throughout the network increases coverage of the search landscape and enables communication of their fittest (nondominated) Abs among each other. Hence, after *negative selection*, jREMISA has the ability to listen for and broadcast all nondominated Abs at the end of each generation, as depicted in Figure 20. System console output confirms the sending and receiving of broadcast Abs. In addition,

users have the ability to broadcast one-line messages to each other, in the event they are utilizing jREMISA in geographically distributed working areas.

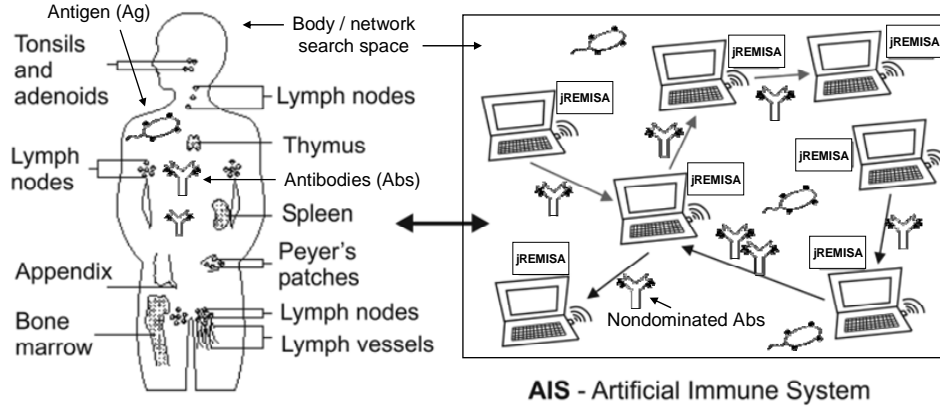


Figure 20: Mapping of BIS distributed components to a distributed AIS

The potential exists to saturate network bandwidth with entire secondary populations being broadcast after every generation. However, each Ab is allowed to be broadcast only once, in its lifetime. Therefore, the user must decide the percentage of the secondary population to broadcast at the end of each generation.

### 3.7 Summary

This Chapter describes the high-level design and specification of our algorithm jREMISA. The methodology addresses the application domain's formal classification and landscape complexity. The foundational EA is multiobjective and fitted with abstract AIS components to make it an AIS-inspired MOEA. The overall algorithm is constructed in order to apply the empirically-derived strengths of REALGO and MISA to our operators of evaluation, selection and mutation. We conclude with how we intend to derive our

approximately optimized solution set through measurable means. Chapter 4 develops our high level design in a more technical depth.

## **IV. Low Level Design and Implementation**

Our jREMISA coding and implementation is a phased project. The end result is one autonomously-operating algorithm that optionally communicates cooperatively with all other computers running jREMISA, within one network segment. This chapter presents the implementation of the high-level Java design and specification details from Chapter 3. Section 4.1 begins with the hardware and software required to perform this software development. Section 4.2 explains the source code migration from C to a prepared software design architecture in Java. Section 4.3 details the signature design, generation of Ab chromosome arrays and the array encoding of the incoming data set A<sub>gs</sub>. Section 4.4 provides pseudocode for all the major functions and evolutionary operators of jREMISA. Section 4.5 explains the distributed communication protocol and transmission methodology of Abs to the subnet of listening jREMISAs. Section 4.6 explains how post-execution data is properly ordered into a saved XML file for graphical analysis and potential future re-use.

The first phase of software implementation involves acquisition and conversion of the existing C programs, REALGO and MISA, into their Java equivalent, jREALGO and jMISA. The second phase involves the integration of the two unrelated Java programs into a single Java program called jREMISA. Once validated for identical output, described in Section 5.2, jREALGO and jMISA's exclusive strengths are merged into a single Java program. The third phase involves tailoring the program to fit the data structure of the real-world data set. Once jREMISA can successfully process external

data files, the fourth and final phase incorporates the distributed component, which facilitates AIS broadcasting of its nondominated Abs among the other AIS' within the LAN.

#### **4.1 Hardware and Software Requirements**

C language analysis and runtime debugging of REALGO and MISA is performed in the Microsoft Visual Studio 2005 Integrated Development Environment (IDE) while Java programming and runtime debugging of jREALGO, jMISA and jREMISA was performed in the *Eclipse*<sup>6</sup> open-source IDE (see Appendix D.3 for source code explanation and hierarchy). To minimize Java execution overhead, compiled Java projects are exported to a self-contained Java Archive File (JAR), executed independently of *Eclipse*, requiring only the Java Virtual Machine (JVM) to be running. Therefore, this program may be executed on any hardware platform running the JVM.

#### **4.2 REALGO and MISA C-To-Java Language Translation**

Because of the growing popularity [Java04], global ubiquity and customer (i.e., DoD) utilization of Java, both REALGO and MISA were converted in Java equivalents jREALGO and jREMISA, respectively, facilitating OS extensibility and flexibility into existing and future customer Java systems. Both REALGO and MISA C language source files were acquired directly from their respective authors and their Java-based derivative are coded to be executed against the same test functions as their C parent.

---

<sup>6</sup> The Eclipse Project, <http://www.eclipse.org>.

Translating REALGO to the Java-based program jREALGO was straightforward because REALGO's signature data structure is a simple bit string array, as Java favors objects over pointers. Translating MISA to the Java-based program jMISA was more difficult due to MISA's chromosome's data *struct(ure)* simply including a pointer to the next chromosome. Hence, C's singly-linked bit string chromosomes became an awkward object-linked Java implementation. In addition, for testing purposes, we “retro-coded” both C programs with a nanosecond-precision timer, made possible with Microsoft Windows' Application Programming Interface (API) system calls (see Appendix D.4). During the translation process, software engineering principles and *design patterns* were incorporated, including one of the earliest, the *Model-View-Controller*, in order to minimize the learning curve for understanding our software development methodology.

#### **4.2.1 The Model-View-Controller Paradigm**

Both REALGO and MISA programs are packaged as the typical “single .C source file with included .H header files.” All parameter values are “hard-wired” into the code, mixing *business logic*, the functions that operate on the program's values, with *program state*, the current value of all parameters defined in the program. From a software engineering standpoint, this maximizes the difficulty level, learning curve and time required of program modification and minimizes software flexibility. Employing the *Model-View-Controller* (MVC) paradigm during the code conversion—one of the earliest known software engineering patterns—helps mitigate this problem (Figure 21).

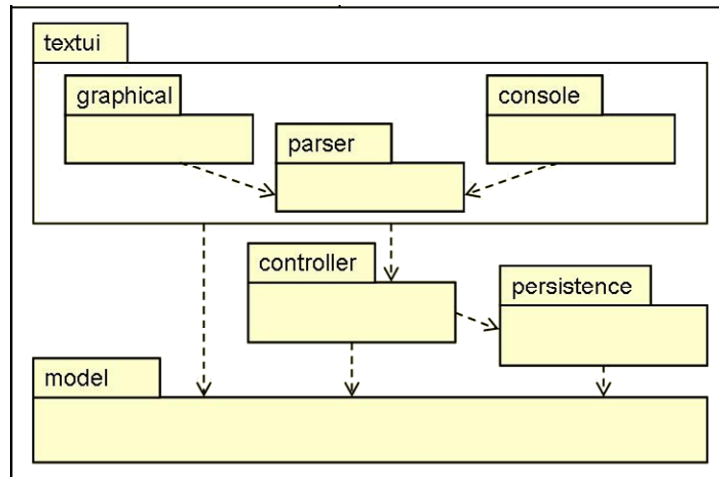


Figure 21: Model-View-Controller architecture [Halloran05]

Per Freeman and Freeman, the *model* “holds all the data, state, and application logic. The model is oblivious to the *view* and *controller*, although it provides an interface to manipulate and retrieve its state and it can send notification of state changes to *observers*” (which the *view* identifies itself as). The *view* “gives you a presentation of the model. The *view* usually gets the state and data it needs to display directly from the model.” And the *controller* “takes user input and figures out what it means to the model” [FF04]. In Figure 21, the arrows represent associated between these *Classes*: the *view* (i.e., “textui”) is aware of the *controller* (and *model*, if required) and the *controller* is aware of the *model* (and *persistence*), but not vice-versa. In this way, the classes can be interchanged with little to modification of the other classes. For example, jREMISA can be fitted with any GUI without changing the *controller* or *model*.

Following this paradigm, C source and header file code are separated into four distinct Java project *packages*:

1. *view*: the graphical user interface (GUI), allowing dynamic parameter selection;
2. *model*: the Java objects that hold state information about the algorithm;
3. *controller*: the mediator that accepts user input (from the GUI) and manipulates state information in the *model*;
4. *persistence*: objects responsible for file input-output, such as reading in a data set file and saving a surviving population into an eXtensible Markup Language (XML) file.

By incorporating the MVC architecture, we have separated the concerns of all files within the Java project. This results in a minimal learning curve and a minimal cost from modification or replacement of any Java *class* within the project *packages*.

### 4.3 Data Signature Design

Per Section 3.6.1, jREMISA employs the bit string data structure, chosen by both REALGO and MISA. Both Abs and Ags are composed of fixed-length integer array chromosomes where *allele* (dimensional) values that define the point's location in the search space are binary and Ab parameter information is of integer value. Ab length is dictated by the Ag length and both signatures are generated in different ways. We begin this discussion with Ag encoding, as this drives Ab generation.



### 4.3.1 Antigen Data Set Encoding

The Abs for network intrusion are generated and trained in the same manner as in anti-virus detectors [HWGL02]. However, network intrusion Ags are longer and segregated because they utilize the IP packet structure for its template. For this reason, we constrain our ID domain to encode Ags from network packets wrapped in the three most common IP protocols: TCP, UDP and ICMP. The encoding process is made possible by a *Java class* called *DumpPro*, courtesy of *SSFNet*<sup>7</sup>. This *class* simply reads in each byte of a binary network traffic file, such as one generated by *tcpdump*<sup>8</sup>, and outputs the decimal values of the various IP fields. This *class* is modified to pre-determine whether the packet was TCP, UDP or ICMP, convert all header fields' decimal values to binary and contiguously concatenate those bit strings into the chromosome that represents the packet header. jREMISA has the flexibility to construct these chromosomes based on user-selected IP, TCP, UDP and ICMP header fields, allowing dynamic re-shaping of the search landscape. Because different authors subjectively choose their chromosome's bit length, as Section 3.1 explained, we must first consider the components of the IP packet, as this determines the length. The user begins jREMISA by selecting the IP, TCP, UDP and ICMP header fields they wish to be evaluated in the search landscape, as shown in Figure 22 for the Massachusetts Institute of Technology (MIT)-Defense Advanced Research Projects Agency (DARPA) 1999 Intrusion Detection data set [MITDARPA99]

---

<sup>7</sup> Scalable Simulation Framework (SSFNet): a clearinghouse for information about the latest tools for scalable high-performance network modeling, simulation, and analysis, <http://ssfnet.org>.

<sup>8</sup> *tcpdump*: network traffic packet analyzer, <http://www.tcpdump.org>.

and Figure 23 for the University of California-Irvine 1999 Knowledge Discovery in Databases (KDD) Cup data set [KDD99]. By default, all fields are chosen.

**Capt HAAG's jREMISA version 0.5**

MAIN | Negative Selection | **Data Structure [MIT-DARPA]** | Data Structure [KDD Cup 99]

4b version	4b header length	8-bit type of service (TOS)	16-bit total length (in bytes)				
16-bit identification			3-bit flag & 13-bit fragment offset				
8-bit TTL			16-bit header checksum				
32b src IP octet A		32b src IP octet B		32b src IP octet C		32b src IP octet D	
32b dest IP octet A		32b dest IP octet B		32b dest IP octet C		32b dest IP octet D	

**IP HEADER**

16-bit source port number		16-bit destination port number	
32-bit sequence number			
32-bit acknowledgement number			
4b header length	6-bit reserved	16-bit window size	
16-bit TCP checksum		16-bit urgent pointer	

**TCP HEADER**

**UDP HEADER**

16-bit source port number		16-bit destination port number	
16-bit UDP length		16-bit UDP checksum	

**ICMP HEADER**

8-bit type	8-bit code	16-bit ICMP checksum
------------	------------	----------------------

Customize your data structure ordering for the MIT-DARPA data set by clicking the IP packet fields you wish to compose your bitstring. By default, all usable fields are currently selected. Smaller-sized headers are padded with 0's to ensure equal length chromosomes. Hover the cursor over a field for its full title.

- WHITE FIELDS: unused
- SOLID FIELDS: selected
- SHINY FIELDS: unselected

CURRENT FIXED MAX BITSTRING LENGTH:  
(IP header + largest size header between TCP, UDP, IP):  
**29 genes, 240 bits (IP=122, TCP=118, UDP=48, ICMP=16)**

Figure 22: jREMISA's MIT-DARPA chromosome construction menu

**Capt HAAG's jREMISA version 0.5**

MAIN | Negative Selection | Data Structure [MIT-DARPA] | **Data Structure [KDD Cup 99]**

duration: length (number of seconds) of the co...	...	service: ne...	flag: normal or en...	src_bytes: numbe...	dst_bytes: numbe...	...	wrong_frag...	urgent: nu...
---------------------------------------------------	-----	----------------	-----------------------	---------------------	---------------------	-----	---------------	---------------

**BASIC FEATURES OF INDIVIDUAL TCP CONNECTIONS**

hot: number of "h...	num_failed...	...	num_comp...	...	num_root: ...	num_file_o...	num_shells: numb...	num_access_files...	num_outbound_o...	...
----------------------	---------------	-----	-------------	-----	---------------	---------------	---------------------	---------------------	-------------------	-----

**CONTENT FEATURES WITHIN A CONNECTION SUGGESTED BY DOMAIN KNOWLEDGE**

count: number of connections to the same hos...	srv_count: number of connections to the same...	error_rate: % ...	srv_error_rat...	error_rate: % ...	srv_error_rat...	same_srv_rat...
diff_srv_rate: ...	srv_diff_host...	dst_host_count	dst_host_srv_count	dst_host_sam...	dst_host_diff...	dst_host_sam...
dst_host_srv...	dst_host_sero...	dst_host_srv...	dst_host_remo...	dst_host_srv...		

**TRAFFIC FEATURES COMPUTED USING A TWO-SECOND TIME WINDOW**

Customize your data structure ordering for the KDD Cup 99 data set by clicking the fields you wish to compose your bitstring. By default, all fields are currently selected. Hover the cursor over a field for its full title.

- WHITE FIELDS: unused
- SOLID FIELDS: selected
- SHINY FIELDS: unselected

CURRENT FIXED MAX BITSTRING LENGTH:  
**41 fields, 291 bits**

Figure 23: jREMISA's KDD Cup 99 chromosome construction menu

While we ideally wish to evaluate all fields, there is a productivity tradeoff between the number of fields chosen and resulting chromosome length. For example, selecting more fields may increase classification effectiveness but certainly increases the search space universe and decrease efficiency. Conversely, fewer fields chosen may reduce the number of *non-self* detected but also the size of the universe, increasing efficiency.

### *IP Packet Background*

The IP packet, shown in Figure 24, has a standardized format where the header is composed of five (rows of) 32-bit *words*, accounting for 20 bytes [Stevens94]. In the first *word* (or row), subtracting *total length* from *header length* results in knowing the *payload* start byte position (with, obviously, the *total length*, itself, denoting the end byte). In the second word, the *identification field* uniquely identifies each IP datagram sent by a host, which increments by one each time a datagram is sent. In the third word, *time-to-live* sets an upper limit on the number of routers a packet can pass through before being dropped; hence, a lifespan. The *protocol* byte is critical in determining which population to route this encoded packet to, as it denotes which protocol (i.e., TCP=6, UDP=17, ICMP=1) gave the data for IP to send. Beyond the header (beginning the 21st byte), the appropriate IP-wrapped protocol packet composes the IP data field, based on the value of the protocol byte. The last two words are the decimal value of the packet's IP source and destination address.

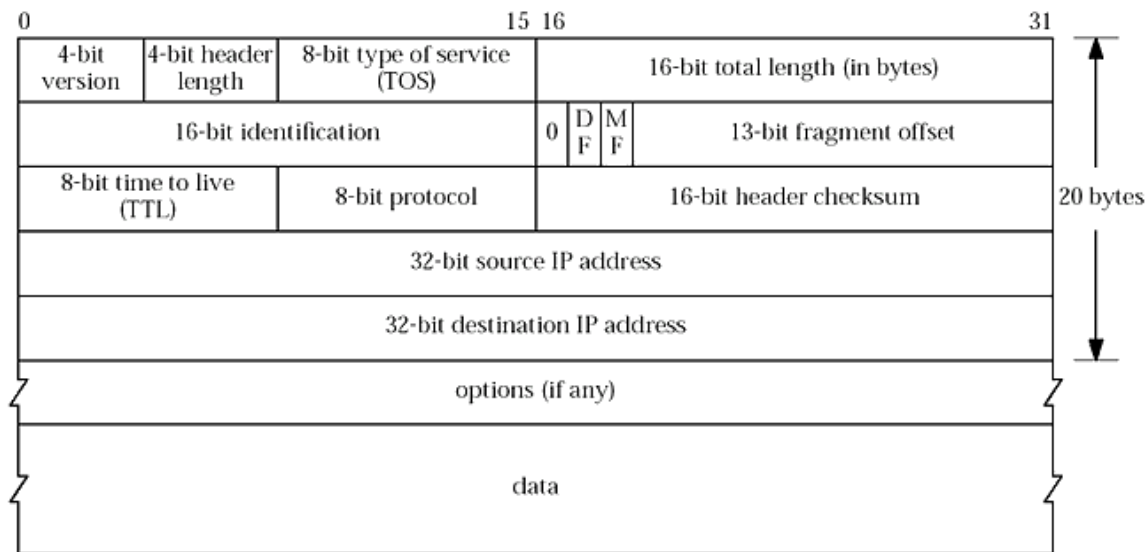


Figure 24: IP datagram packet [Stevens94]

If the IP protocol byte value equals six, then we conclude this IP packet is more specifically a TCP/IP—or “TCP over IP”—packet; in other words, the 21st byte of this IP packet is the first byte of the TCP header. The TCP packet, shown in Figure 25, also has a fixed header of 20 bytes with a variable byte-size payload. The first word contains the *port* (or *service*) number of this packet. The second word assists in ensuring TCP packets are read in the correct order received as sent, as packets can traverse varying routes and arrive at different times, possibly out of order. The third word is the value of the second word plus one, sent back to the original sender, confirming to that sender his packet was received. In the fourth word, subtracting this TCP *header length* and IP header length from the IP total length yields the TCP payload size and starting byte. The six Boolean flag bits, URG, ACK, PSH, RST, SYN and FIN assist in packet control and connection setup and teardown.

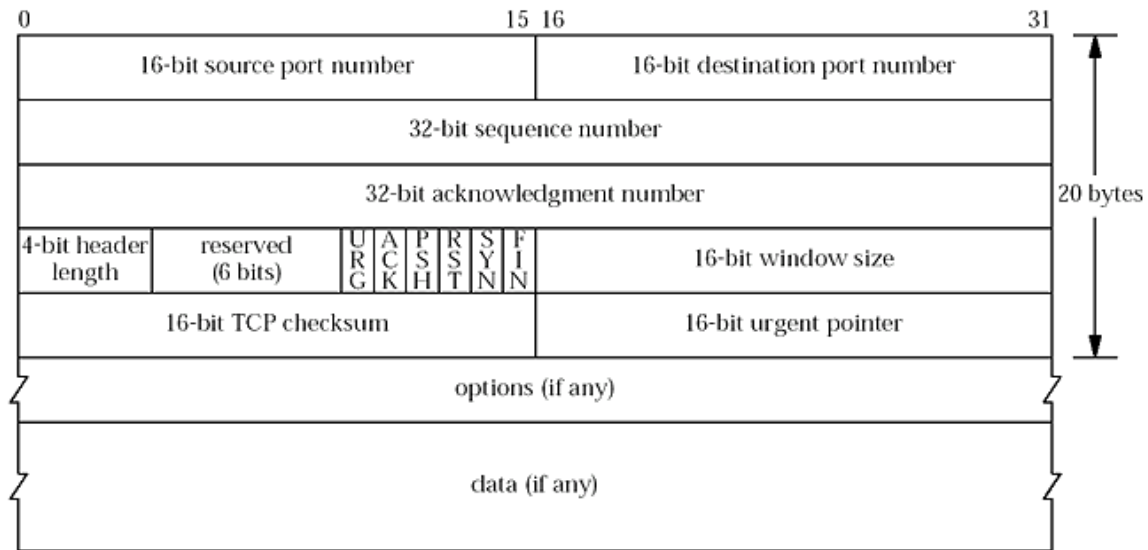


Figure 25: TCP packet [Stevens94]

If the IP protocol byte value equals 17, then we conclude this IP packet is more specifically a UDP/IP—or “UDP over IP”—packet; in other words, the 21st byte of this IP packet is the first byte of the UDP packet (Figure 26). This protocol’s header is only eight bytes because it doesn’t require (i.e., it’s not responsible) for ensuring successful end-to-end transmission (or *stateful session*)—this is a “fire and forget” *stateless* protocol. The first word contains the source and destination ports. In the second word, we are concerned only with half-word *UDP length*, as subtracting this from the *UDP header* length yields the UDP payload size and start byte.

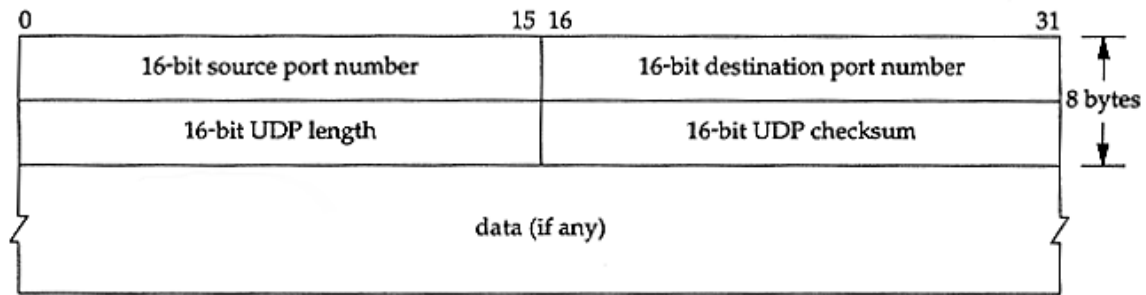


Figure 26: UDP packet [Stevens94]

If the IP protocol byte value equals one, then we conclude this IP packet is more specifically a ICMP packet; in other words, the 21st byte of this IP packet is the first byte of the ICMP packet, shown in Figure 27. In the first word, the first two bytes determine the type of message this is, which is detailed after the first word for a variable size.

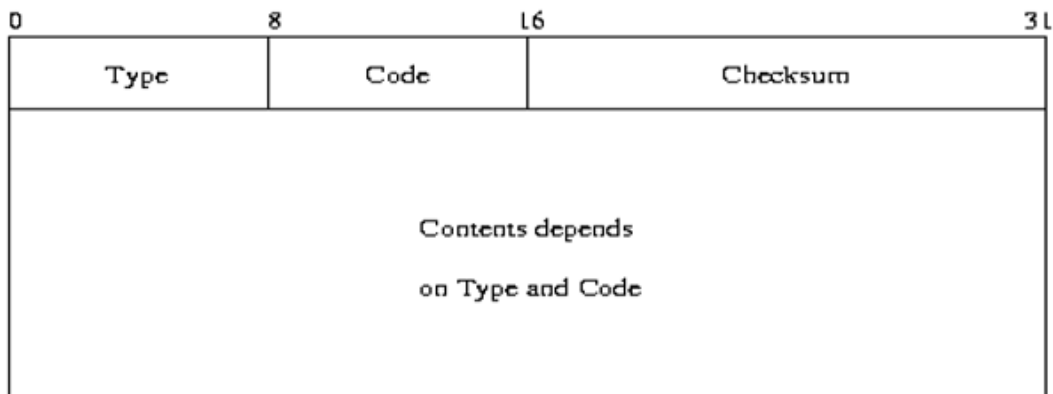


Figure 27: ICMP packet [Stevens94]

Each component selected for the Ag is individually encoded from decimal into a binary *gene*—a *building block* or subset of bits of our chromosome. All genes contiguously aligned by order of field compose one Ag chromosome. This idea of Ag

encoding came from Harmer and Williams whose chromosome length was 320 bits, leaving the trailing 11 bits for payload consideration [HWGL02, Williams01]. Our encoding scheme differs by disregarding payload, several IP and TCP fields and the *validity bit*, yielding Ags ranging in size between 138-240 bits, depending on the underlying protocol (Table 3).

Gene #	Field	Possible Values	Start Loc	Gene Bits	Comment
<i>IP Field: Common to all packets</i>					
1	IP overall packet length	0-65535	0	16	
2	datagram identification number	0-65535	16	16	
3	3-bit flag & 13-bit fragment offset	0-65535	32	16	Values 2, 4 possible and legal; all others suspect [Williams01]
4	time-to-live (TTL)	0-255	48	8	
5	Protocol type	1 (TCP), 2 (UDP), 3 (ICMP)	56	2	“0” corresponds to IP packets not of the underlying TCP, UDP, or ICMP protocol; by forcing TCP=1 (originally 6), UDP=2 (originally 17) and ICMP=3 (originally 1), we shorten number of bits from 8 to 2
6	IP Src Address A octet	0-255	58	8	
7	IP Src Address B octet	0-255	66	8	
8	IP Src Address C octet	0-255	74	8	
9	IP Src Address D octet	0-255	82	8	
10	IP Dst Address A octet	0-255	90	8	
11	IP Dst Address B octet	0-255	98	8	
12	IP Dst Address C octet	0-255	106	8	
13	IP Dst Address D octet	0-255	114	8	
<i>TCP Fields</i>					
T14	TCP source port	0-65535	122	16	
T15	TCP destination port	0-65535	138	16	
T16	TCP Sequence number	0-4294967295	154	32	
T17	TCP Ack number	0-4294967295	186	32	
T18	TCP URGent flag	0-1	218	1	1 = set, 0 = not set
T19	TCP ACK flag	0-1	219	1	
T20	TCP PuSH flag	0-1	220	1	
T21	TCP ReSeT flag	0-1	221	1	
T22	TCP SYN flag	0-1	222	1	

T23	TCP FINish flag	0-1	223	1	
T24	TCP window size	0-65535	224	16	
	TCP data		240		Not currently used
<i>UDP Fields</i>					
U14	UDP Src port	0-65535	122	16	
U15	UDP Dst port	0-65535	138	16	
U16	UDP length	0-65535	154	16	
	UDP data		170		Not currently used
<i>ICMP Fields</i>					
I14	ICMP type	0-255	122	8	
I15	ICMP code	0-255	130	8	
	ICMP checksum	0-65535	138	16	Variable, depending on type and code; not currently used
	ICMP data				Not currently used

Table 3: Antibody signature design [adapted from HWGL02]

An example of an encoded TCP DNA Ag chromosome is depicted in Figure 28. Here, all IP and TCP fields have been selected, resulting in a 240-bit chromosome, which dictates the initialized TCP Ab population to be of fixed length 240 bits, as well.

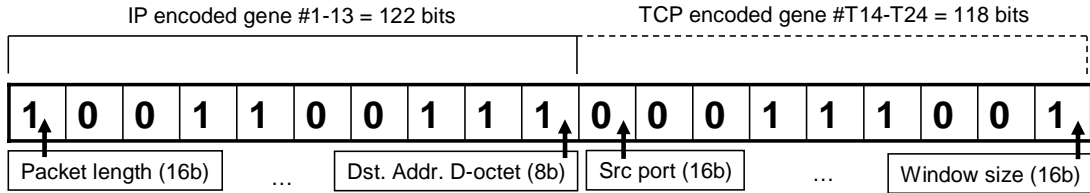


Figure 28: Example encoding of a IP and TCP header to form a TCP DNA chromosome

#### 4.3.2 Antibody Population Generation

When Ag chromosome encoding is determined, the fixed-length TCP, UDP and ICMP Ag are known: a TCP Ag =  $|\text{IP}| + |\text{TCP}|$ , a UDP Ag =  $|\text{IP}| + |\text{UDP}|$  and a ICMP Ag =  $|\text{IP}| + |\text{ICMP}|$ . These three values, along with the user-defined size of the TCP, UDP and ICMP Ab populations, provide the information required to perform Ab *initialization*.



Three individual Java *ArrayLists*, representing the population pools, are instantiated to hold each created Ab integer array. An Ab is composed of three contiguous parts:

1. its chromosome of binary values, defined through negative selection;
2. its RNA memory, initialized to zero;
3. seven parameters that define the state of the Ab, all initialized to zero:

$\lambda \leftarrow$  name (integer identifier);

$\alpha \leftarrow$  number of false detections;

$\rho \leftarrow$  (true positive + true negative) fitness score;

$\phi \leftarrow$  (false positive + false negative) fitness score;

$\eta \leftarrow$  deviation from *negative selection*-defined affinity threshold (determining volume);

$\beta \leftarrow$  broadcasted (0=no, 1=yes; only happens once in its lifetime);

$\psi \leftarrow$  number of Abs that *Pareto*-dominate this Ab.

When Abs are first generated at *negative selection*, their data structure is only the first part—its chromosome—because the RNA memory and parameters serve no purpose. These latter two parts are attached to each Ab upon loading them into their respective populations for post-*negative selection* evaluation. An example of a TCP Ab chromosome is shown in Figure 29. Here, this Ab’s RNA matches its chromosome, is named “3”, has two false detections, a true detection fitness of 38, a false detection fitness of 126, an affinity shrinking of 2%, has been broadcasted and is dominated by two other Abs.

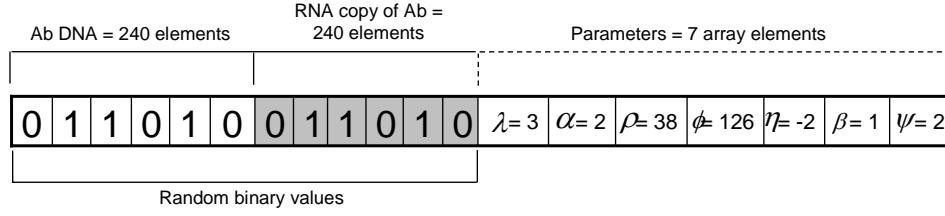


Figure 29: Example 240 bit (487-element) TCP Ab chromosome

A chromosome of length 240 is really 487 elements long because the RNA is the same length as the DNA and then eight state parameters follow. This does not result in additional computational cost as only the DNA is ever computed upon, allele by allele.

#### 4.4 AIS-Inspired MOEA Pseudocode

Section 3.3 introduced the layered complexity involved in designing an AIS-inspired MOEA. Figure 30 has been adapted from Timmis' AIS abstract model (from Section 2.3) to represent jREMISA's integration. The foundation is based on the ID domain, facilitated by the data set introduced in Chapter 5.3. Ab and Ag data structure is the integer array, where the DNA and RNA is composed of zeros and ones and the parameters are signed integer values. Our affinity measure is Hamming distance. The immune algorithm employed is an integration of REALGO and MISA operators, minus recombination, within a Bäck EA ordering.

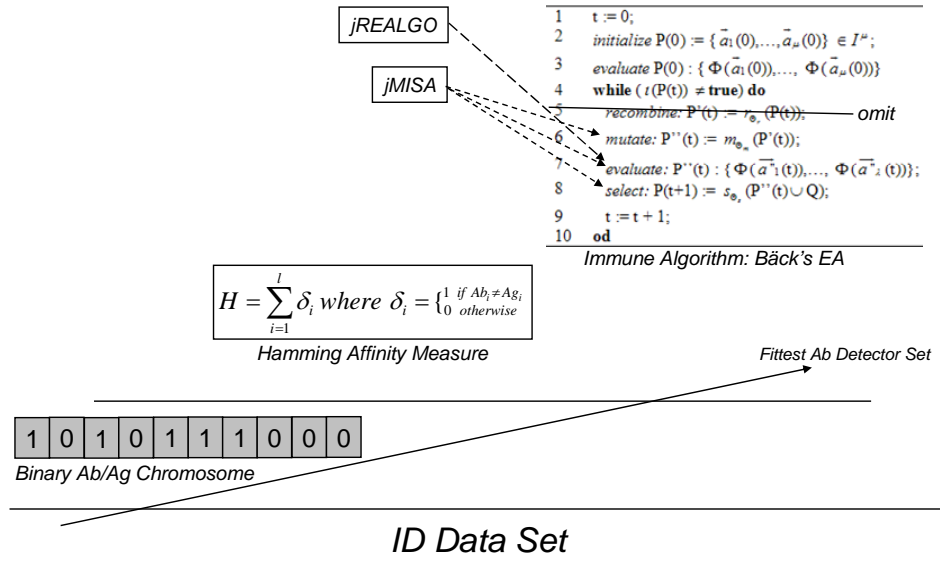


Figure 30: jREMISA applied to Timmis' AIS abstract model

This defined framework provides enough information to map our operators into an ordered sequence, depicted in Figure 31, subsequently enabling pseudocode to be generated, as shown in Algorithm 3. This algorithm changes the order of Bäck's EA within the *while* loop by placing evaluation first instead of last. This is due to the influence of the original MISA algorithm order. While Algorithm 3 shows jREMISA as a single algorithm of execution, jREMISA actually executes in two separate phases: “Phase I: negative selection” (Algorithm 3, lines 3-7) and “Phase II: core MOEA” (Algorithm 3, lines 8-19), because while both operations compose the MOEA, jREMISA halts upon completion of the first phase. This action allows the jREMISA operator to consider if the trained-and-immature population is worthy of being input into the rest of the MOEA.

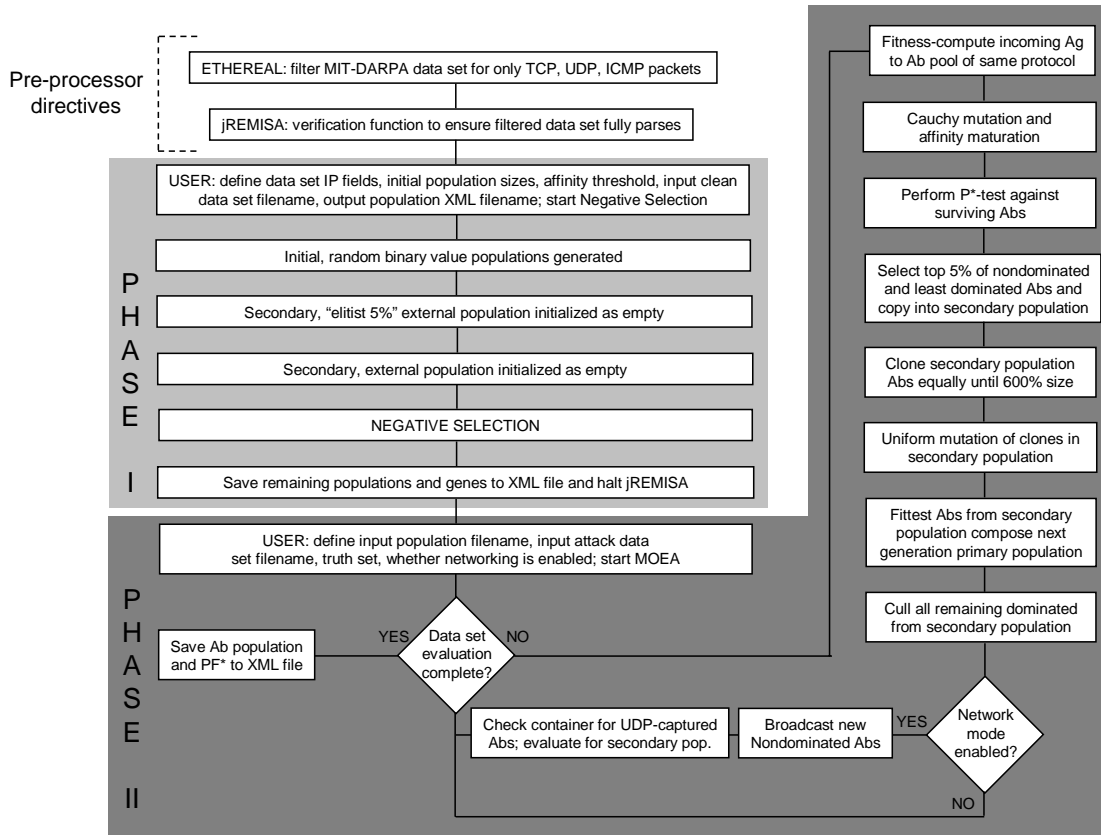


Figure 31: jREMISA algorithm flowchart

---

```

1  procedure jREMISA
2  begin
3  repeat
4      Randomly generate initial TCP, UDP, ICMP Populations ( $P_p$ )
5      Initialize empty secondary Population ( $P_s$ )
6      negative_selection( $P_p$ , data_setclean, threshold) /* Evaluation 1 */
7  until (end of data_setclean)
8  repeat
9      fitness function (ag) /* evaluation_2 */
10     mutationCauchy( $P_p$ )
11     P_optimality() /* evaluation_3 */
12     clonalSelection(0.05)
13     mutationUniform( $P_s$ )
14      $P_p \leftarrow P_s$  /* copy best of  $P_s$  as next gen's  $P_p$  */
15     if (networking)
16         broadcast( $P_s$ ) /* offer nondominateds to other AISs */
17         processReceived() /* Any captured Abs from others? */
18     endif
  
```

---

---

```

19 until (end of data_setattack)
20 end

```

---

Algorithm 3: jREMISA pseudocode

#### 4.4.1 Phase I: Negative Selection

*Negative selection* involves three parameters: the populations to train, the clean data set that trains the populations, and the *affinity threshold*. The random values that initialize the chromosome alleles are determined by Java's native *Random* class. The number of generations of execution is equal to the number of packets in the *self*-only data set. Based on the *affinity threshold* parameter set by the user, all Abs that react to the *self* Ag comparator are removed from the population, without replacement. We did this rather than retrain the Ab through mutation because of the possibility a trained Ab may now match at least one *self* Ag previously tested. Upon completion of this phase, jREMISA halts, awaiting further input. Pseudocode for this phase is given in Algorithm 4.

---

```

1 procedure negative_selection(Pi,data_setclean,threshold)
2 begin
3 repeat
4   ag ← encode_Ag(data_set_packet)
5   for (i ← 0 to size(Pi)) /* Pi ← random, initial population */
6     ab ← Pii
7     score ← HammingScore(ab,ag)
8     if (score ≥ threshold)
9       remove(Pii)
10    endif
11  next (i)
12 until (data_set = end_of_file)
13 end

```

---

Algorithm 4: jREMISA negative selection pseudocode

#### 4.4.2 Phase II: Core MOEA

This phase, given the population and several other user-defined parameters, executes the remainder of MOEA algorithm. Per Algorithm 3, this phase performs the *fitness function*, *Pareto optimality*, *mutation*, *selection* and nondominated Ab-broadcasting (if enabled) upon each Ab, where the number of generations is equal to the number of data set network packets.

##### *Fitness Function*

The *fitness function* (Algorithm 5) calculates the number of false detections, true and false detection integer fitness and affinity threshold deviation value for each Ab. Our fitness scoring method is based on the fitness scoring model conceived by Smith Forrest and Perelson in their search for diverse, cooperative populations with GAs [SFP93] but tailored to our MOEA. The RNA storage and reversion functions and Cauchy mutation method are REALGO-inspired. To aid the *fitness function*, a Java *TreeSet* stores all extracted *non-self* packet identifying numbers in ascending order to guide the outcome of the Ab classification. The *non-self* packet determination methodology is further discussed in Appendix B.

---

```
1  procedure fitness_function(Pi,data_set,threshold)
2  begin
3  repeat
4    ag ← encode_ag(data_set_packet)
5    for (i ← 0 to size(Pi)) /* Population of same protocol as Ag */
6      ab ← Pi
7      (H,HammingMask) ← HammingScore(ab,ag)
8      score ← (H / length(ag))
9      if (ag == self && score < threshold) /* true neg */
```

---

---

```

10       $ab_{fitness1} \leftarrow ab_{fitness1} + H$ 
11       $ab_{RNA} \leftarrow ab$ 
12       $ab_{threshold} \leftarrow ab_{threshold} + 1$ 
13       $mutateAllele \leftarrow 1$ 
14      else if ( $ag == non-self \ \&\& \ H \geq threshold$ ) /* true pos */
15           $ab_{fitness1} \leftarrow ab_{fitness1} + (length(ag) - H)$ 
16           $ab_{RNA} \leftarrow ab$ 
17           $ab_{threshold} \leftarrow ab_{threshold} + 1$ 
18           $mutateAllele \leftarrow 0$ 
19      else if ( $ag == self \ \&\& \ H \geq threshold$ ) /* false neg */
20           $ab_{falseDetections} \leftarrow ab_{falseDetections} + 1$ 
21          if ( $ab_{falseDetections} == lifespan$ )
22               $remove(ab)$ 
23              break
24          end if
25           $ab \leftarrow ab_{RNA}$ 
26           $ab_{fitness2} \leftarrow ab_{fitness2} + H$ 
27           $ab_{threshold} \leftarrow ab_{threshold} - 1$ 
28           $mutateAllele \leftarrow 1$ 
29      else if ( $ag == non-self \ \&\& \ H < threshold$ ) /* false pos */
30           $ab_{falseDetections} \leftarrow ab_{falseDetections} + 1$ 
31          if ( $ab_{falseDetections} == lifespan$ )
32               $remove(ab)$ 
33              break
34          end if
35           $ab \leftarrow ab_{RNA}$ 
36           $ab_{fitness2} \leftarrow ab_{fitness2} + (length(ag) - H)$ 
37           $ab_{threshold} \leftarrow ab_{threshold} - 1$ 
38           $mutateAllele \leftarrow 0$ 
39      end if
40       $ab \leftarrow CauchyMutation(ab, HammingMask, mutateAllele)$ 
41  next (i)
42  until (data_set = end_of_file)
43  end

```

---

Algorithm 5: jREMISA fitness function pseudocode

The Hamming equation serves two purposes: sum the number of complimentary bits as Hamming score  $H$  and mark the allele positions of complementarity with a “1” value, in what we developed as a “Hamming mask.” For any outcome, there is a penalty

added to the true classification (called  $\text{fitness}_1$ ) or false detection fitness (called  $\text{fitness}_2$ ) scores involving the  $H$  value. The Hamming mask is the heuristic that mutation uses to determine which alleles to mutate. There are four possible outcomes for each Ab, each having a unique consequence:

1. the Ag is *self* and the Ab declares *self* (true negative):
  - a. add  $H$  to  $\text{fitness}_1$ , penalizing one point for every allele of complementarity;
  - b. save DNA chromosome as its RNA for correctly classifying the Ag;
  - c. increment (reward) Ab's affinity deviation by one, enlarging its volume;
  - d. mark the Hamming mask alleles with a "1" for Cauchy mutation because there should not have been complementarity between two *self*s.
2. the Ag is *self* and the Ab declares *non-self* (*Type-II* error: false negative):
  - a. increase the "false detections" counter by one (and remove from population if false detection threshold reached, bypassing the remaining operations);
  - b. restore Ab DNA chromosome with its RNA, as it was more effective than this Ab's mutation from last generation;
  - c. add  $H$  to  $\text{fitness}_2$ , penalizing one point for every allele of complementarity;
  - d. decrement (penalize) Ab's affinity deviation by one, shrinking its volume;
  - e. mark the Hamming mask alleles with a "1" for Cauchy mutation because there should not have been complementarity between two *self*s.



3. the Ag is *non-self* and the Ab declares *non-self* (true positive):
  - a. add opposite of Hamming score ( $Ag_{\text{length}} - H$ ) to  $\text{fitness}_1$  because there should have been more complementarity;
  - b. save DNA chromosome as its RNA for correctly classifying the Ag;
  - c. increment (reward) Ab's affinity deviation by one, enlarging its volume;
  - d. mark the Hamming mask alleles with a "0" for Cauchy mutation because there should have been more complementarity between *self* and *non-self*.
4. the Ag is *non-self* and the Ab declares *self* (*Type-I* error: false positive):
  - a. increase the "false detections" counter by one (and remove from population if false detection threshold reached, bypassing the remaining operations);
  - b. restore Ab DNA chromosome with its RNA, as it was more effective than this Ab's mutation from last generation;
  - c. add opposite of Hamming score ( $Ag_{\text{length}} - H$ ) to  $\text{fitness}_2$  because there should have been more complementarity;
  - d. decrement (penalize) Ab's affinity deviation by one, shrinking its volume;
  - e. mark the Hamming mask alleles with a "0" for Cauchy mutation because there should have been more complementarity between *self* and *non-self*.

When a Ab's chromosome is reverted to its RNA upon a false detection, the remaining parameters of the Ab are unchanged; in other words, parameters do not carry when an RNA copy of an Ab is made. Having the Hamming mask and the alleles to target now allows us to perform heuristic-based Cauchy mutation. Each Ab has its own

volume, defined by its threshold deviation parameter added to the *negative selection*-defined affinity threshold value.

### *Heuristic-based Cauchy-mutation*

While REALGO gives each allele a 50% chance of Cauchy mutation (see Section 3.6.3), we heuristically determine the number and position of alleles to mutate based on our Hamming mask. As just discussed, Abs that did not have enough complementarity are penalized for the remaining alleles not complementary ( $\text{mutateAllele} \leftarrow 0$ ). Conversely, Abs that should not have experienced complementarity are penalized for the alleles that were complementary to the Ag ( $\text{mutateAllele} \leftarrow 1$ ). Ab alleles are mutated in this manner because the alleles that are properly complementary should not change value, as shown in Figure 32.

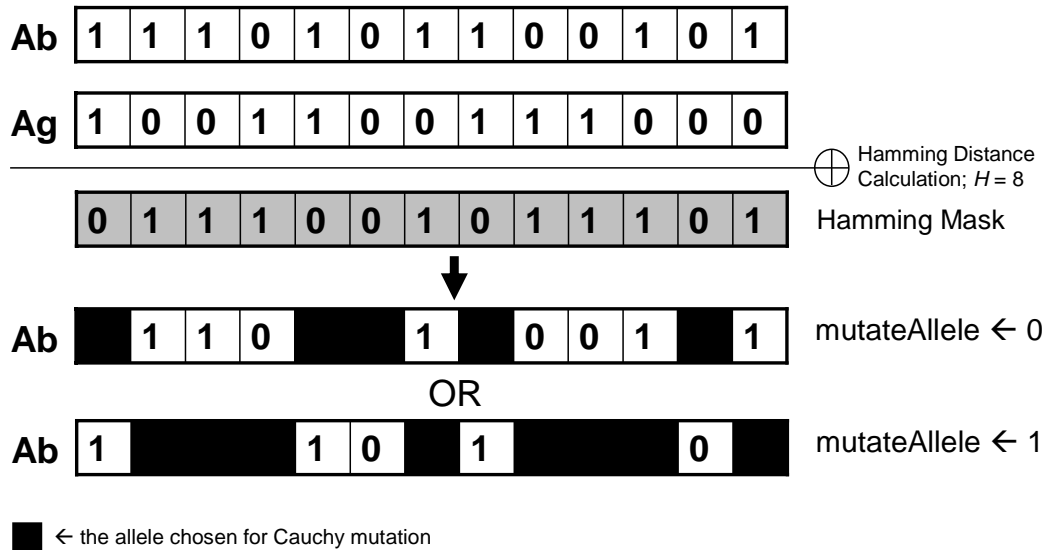


Figure 32: Allele selection process for Cauchy Mutation

### Clonal Selection

Although the top 5% of primary population-Abs are copied into the secondary population, with the intent of being cloned, only those Abs new to the secondary population are cloned. The “name” parameter on each Ab enables this determination. Upon cloning completion, *Quicksort* is applied to put the secondary population in “number of Abs dominated by”-ascending order in order to minimize the time required to copy enough of the fittest Abs from the secondary population to restore the original size of the primary population (if required, in the event any Abs were lost from the primary population due to excessive false detections). The *selection* pseudocode is shown in Algorithm 6.

---

```
1  procedure selection(0.05,popp,pops)
2  begin
3  repeat /* Copy top 5% to secondary pop */
4      copyToSecondary(popp.get(i))
5      i ← i + 1
6  until (i == 0.05*size(popp))
7  numClones ← (size(pops)*6 / i) /* Num clones per Ab */
8  repeat /* uniformly clone Abs to 600% pops size */
9      ab ← popp.get(i)
10     j ← numClones
11     repeat
12         abc ← copy(ab) /* clone if new to secondary pop */
13         mutation(abc, abdominated_score)
14         copyToSecondary(abc) /* insert into pops */
15         j ← j - 1
16     until (j == 0)
17     mutation(ab, abdominated_score) /* mutate original ab, too */
18     i ← i - 1
19 until (i == 0)
20 QuickSort(pops) /* ascending sort of Abs by “dominated” score */
21 popp ← copyToPrimary(pops, size(popp)) /* evolution:  $\mu + \lambda$  */
22 i ← size(pops)
```

---

---

```

23 repeat /* Cull pops to nondom-only, no larger than primary pop */
24     ab ← pops.get(i)
25     if (abdominated_score > 0)
26         remove(pops,ab)
27     end if
28     i ← i - 1
29 until (i == 0)
30 end

```

---

Algorithm 6: jREMISA selection pseudocode

#### 4.5 Distributed Communication Model

Our distributed communication model is based on Grama, Gupta, Karypis and Kumar’s definition of *data decomposition* [GGKK03]. We partition a particular day’s data set into  $c$  equal sizes, where  $c$  is number of computers executing jREMISA. Hence, each jREMISA is evaluating an equal portion of the data set in a *distributed island model*, broadcasting nondominated Abs to each other’s secondary population in an effort to synergistically strengthen effectiveness in the AIS system, as a whole subnet. Hence, our intent for pursuing distributed execution is geared more toward increasing effectiveness than the expected  $c$ -fold increase in efficiency.

To facilitate communication, jREMISA binds to one UDP port for listening and one for broadcasting. The listener *class* must spawn its own Java *Runnable* class thread of concurrent execution because it blocks execution until receiving information from the broadcast port. Each message, whether a nondominated Ab or user message, is sent in a single UDP packet to IP address 255.255.255.255, where it is summarily broadcast to all jREMISA listeners on the subnet, as depicted in Figure 33.

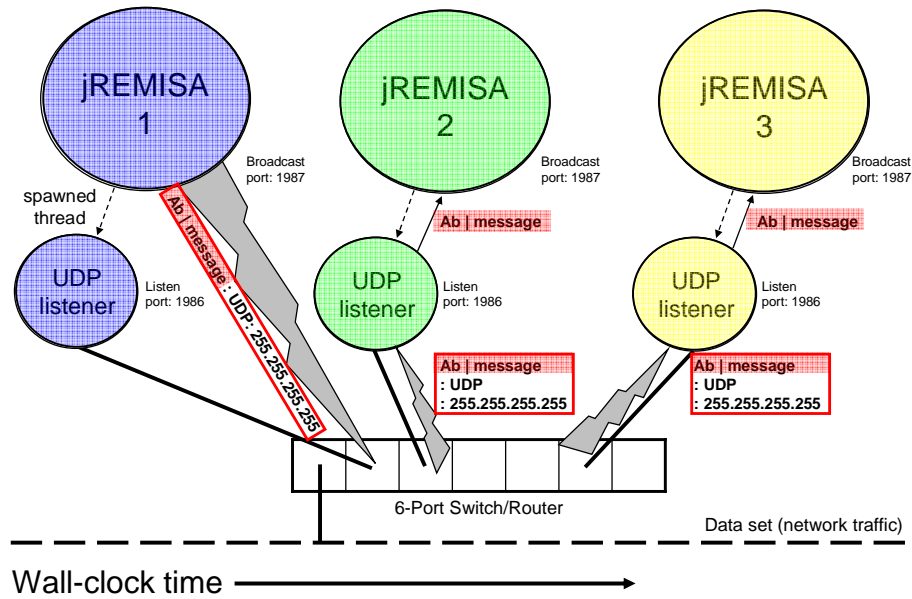


Figure 33: jREMISA distributed communication architecture

In Figure 33, jREMISA-1 sends a newly discovered nondominated Ab, wrapped in a UDP packet, to IP address 255.255.255.255. All jREMISA listener threads capture the message, unwrap it and send the payload content to its respective kernel, where it is examined only after a generation ends. Ab payloads captured by the listeners are *Pareto optimality*-evaluated against their secondary population and added if that Ab remains nondominated. Abs are broadcast only once in their lifetime. Java *synchronized methods* and *volatile* variables are employed to ensure thread-safe passage of broadcast Abs into the dynamically-changing secondary population. Instant messages are simply received and sent to the GUI display console. The UDP payload, depicted in Figure 34, consists of the sender IP address, machine hostname, message type and the user message or Ab integer array.

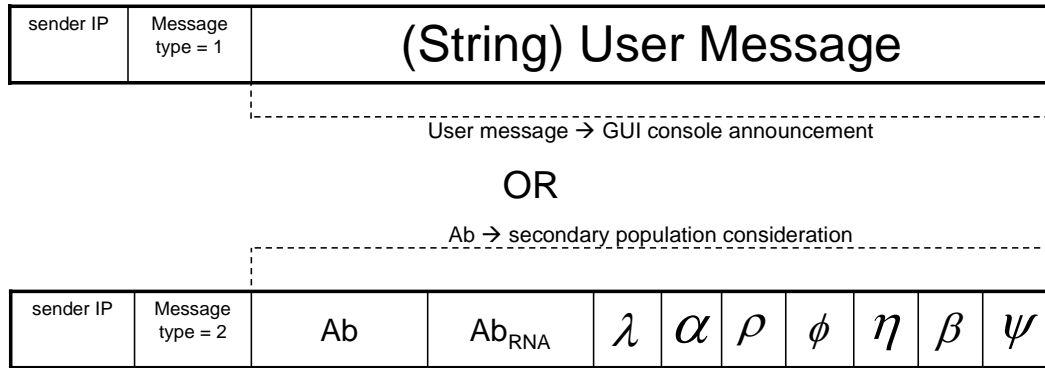


Figure 34: UDP-broadcast payload structure

#### 4.6 Population Persistence

Software persistence involves the long-term storage of data, for future reference. For example, volatile storage involves an algorithm’s execution and data manipulation in memory; once power is lost or the algorithm terminates, the memory is lost. Non-volatile storage persists data to long-term storage mediums such as hard disk and removable, flash-memory “thumb drives.” While our algorithm outputs the *Pareto optimal* values at post-execution, we desire to know the Ab data structure behind that set of values for analysis and future ID domain employment. Our algorithm preserves post-execution output in XML format based on the executed function:

1. **NEGATIVE SELECTION:** saves the user-defined data structure and trained-but-immature Ab populations (Figure 35);
2. **Core MOEA:** saves the user-defined data structure, number of generations, runtime in seconds, elitism percentage, affinity threshold, true classification rate, false detection rate, attack graph x- and y-vectors, and each secondary

population's set of Abs with accompanying *Pareto Front* x- and y- vectors (see Section D.2.3, Figure 60).

```
<?xml version="1.0" encoding="UTF-8"?>
<jREMISA version="0.5">
  <dataset type="MIT_DARPA99" HammingAffinityThresholdPercent="0.25">
    <genefield name="1" bitlength="16" descrip="16-bit total length (in bytes)" />
    <genefield name="2" bitlength="16" descrip="16-bit identification" />
    <genefield name="3" bitlength="16" descrip="3-bit flag & 13-bit fragment offset" />
    <genefield name="4" bitlength="8" descrip="8-bit TTL" />
    <genefield name="5" bitlength="2" descrip="8-bit protocol (only need two lowest bits)" />
    <genefield name="6" bitlength="8" descrip="32b src IP octet A" />
    <genefield name="7" bitlength="8" descrip="32b src IP octet B" />
    <genefield name="8" bitlength="8" descrip="32b src IP octet C" />
    <genefield name="9" bitlength="8" descrip="32b src IP octet D" />
    <genefield name="10" bitlength="8" descrip="32b dest IP octet A" />
    <genefield name="11" bitlength="8" descrip="32b dest IP octet B" />
    <genefield name="12" bitlength="8" descrip="32b dest IP octet C" />
    <genefield name="13" bitlength="8" descrip="32b dest IP octet D" />
    <genefield name="14" bitlength="16" descrip="16-bit source port number" />
    <genefield name="15" bitlength="16" descrip="16-bit destination port number" />
    <genefield name="16" bitlength="32" descrip="32-bit sequence number" />
    <genefield name="17" bitlength="32" descrip="32-bit acknowledgement number" />
    <genefield name="18" bitlength="1" descrip="URGent" />
    <genefield name="19" bitlength="1" descrip="ACKnowledgement" />
    <genefield name="20" bitlength="1" descrip="PUSH" />
    <genefield name="21" bitlength="1" descrip="RESET" />
    <genefield name="22" bitlength="1" descrip="SYNchronize" />
    <genefield name="23" bitlength="1" descrip="FINish" />
    <genefield name="24" bitlength="16" descrip="16-bit window size" />
    <genefield name="UL4" bitlength="16" descrip="16-bit source port number" />
    <genefield name="UL5" bitlength="16" descrip="16-bit destination port number" />
    <genefield name="UL6" bitlength="16" descrip="16-bit UDP length" />
    <genefield name="IL4" bitlength="8" descrip="8-bit type" />
    <genefield name="IL5" bitlength="8" descrip="8-bit code" />
  </dataset>
  <TCPantibodies numAbs="5" bitlength="124">
    <Ab chromosome="000101010000111111110100001100010001011111010001000001000100101110101101111111011010110001110011" />
    <Ab chromosome="110100001000111100100011011101111100111101101010101100011000100001010001001011010110110110111" />
    <Ab chromosome="010100100100100100000001001010100100110001101101110110010010000001001010001101011011011011011" />
    <Ab chromosome="10000001000001001010110101110001010001000110010010110011011011011011011011011011011011011011011" />
    <Ab chromosome="1001000100000001111010110001010111011101101010010001010101111101000101110100101010110110110110111" />
  </TCPantibodies>
  <UDPantibodies numAbs="4" bitlength="98">
    <Ab chromosome="111111011001100000011111000101000110010001111000101000110101001000101110101011111101011001000" />
    <Ab chromosome="001110110000000001010010000010010111011110111101101110000101101010101101110000100010111000" />
    <Ab chromosome="110001001100100100100000110110001100011111011011000110001101111010000001101100110101010" />
    <Ab chromosome="0001110101000101011001011000100110011001101001101001011010010110110110110000101110111" />
  </UDPantibodies>
  <ICMPantibodies numAbs="3" bitlength="82">
    <Ab chromosome="111101010010010011010001100111100011001000101101010111001000100001000100011100" />
    <Ab chromosome="01111001110001000101110010010001100100110010001110001010001010001010011111100" />
    <Ab chromosome="0100010111001001011011110111011010000110101101110011000100000011010001101" />
  </ICMPantibodies>
</jREMISA>
```

Figure 35: Example XML post-negative selection file

XML files serve as a “Petri dish,” enabling population re-use. For example, if the user wants to perform negative selection over five different data sets before applying the attack set, the saved post-negative selection XML file can be reloaded for continued negative selection over four additional times before specified as the input file for the attack set evaluation. In addition, post-MOEA Petri dishes allow for trained-and-mature population re-use in further ID domain evaluations.

## **4.7 Summary**

This Chapter discusses our low-level design and software implementation plan in order to prosecute problem domain input. The next chapter discusses the testing and experimentation performed using this software and the analysis of our computational results.



## **V. Experimentation and Analysis**

The previous two chapters discussed the high- and low-level methodology for complete prototype implementation. This chapter presents the experimentation and analysis plan intended to produce results which can be evaluated against our hypotheses objectives and other algorithms employing the same data set and experiments. Section 5.1 begins by describing our testing environment and objectives, with background on the test functions and data sets used in facilitating these tests. Section 5.2 provides validation for the migration of REALGO and MISA to jREALGO and jMISA. Section 5.3 validates jREMISA against the benchmark ID data set. Section 5.4 then compares our work to others who have applied this data set as their application domain. Section 5.5 summarizes by reviewing the outcome of our experiments and how it impacted our hypothesis objectives.

### **5.1 Experimental Objectives and Design**

The purpose of these experiments is to determine if an AIS-inspired MOEA is useful in effectively classifying network events while its Abs maintain an optimally known hypervolume. The experimental results provide the measurements that our hypothesis objectives require in order to declare whether our conjecture is valid. Our experiments are divided into three parts:

1. validation of the C-to-Java algorithm migration through test functions;
2. measuring the effectiveness of jREMISA by evaluating the ID domain data set in 13 different scenarios:

- a. 10 standalone execution scenarios, involving at least one evaluation of each day of an entire week of attacks;
  - b. three *distributed island model* executions in a two-, three- and four-jREMISA configuration;
3. determining jREMISA's worth against other algorithms applied to the same problem domain through statistical analysis.

### 5.1.1 Testing Environment

Algorithm evaluation is conducted in two configurations: standalone and *distributed island model*, involving the following computers, which we identify by name:

1. "PC1" ← Dell Inspiron 710m laptop, 2.0 GHz Pentium M, Intel Centrino processor, two GB of RAM, Windows XP Professional 2002, Service Pack 2;
2. "PC2" ← Dell XPS laptop, 3.4 GHz Pentium 4 Hyper-Threading processor, one GB of RAM, Windows XP Professional 2002, Service Pack 2;
3. "PC3" ← Dell Precision laptop, 1.8 GHz Pentium 4 processor, one GB of RAM, Windows XP Professional 2002, Service Pack 2;
4. "PC4" ← Dell Optiplex GX270 workstation, 2.6 GHz Pentium 4 processor, 512 MB of RAM, Windows XP Professional 2002, Service Pack 2.

The standalone configuration employed PC1. The distributed phase involved all four machines connected via *Category-5* network patch cables to a Cisco 4-port wireless router transmitting at 100 mbps. In order to take advantage of as much of the machine's memory as possible, the *Eclipse*-exported jREMISA JAR is executed independent of *Eclipse* in the Windows *COMMAND PROMPT* with the command line argument, "java –

XX:+AggressiveHeap -jar jREMISA.jar”. In doing this, we observed Windows *Task Manager* reporting jREMISA utilizing 0.7 GB of virtual memory, 220 MB of physical RAM and 95% CPU usage during execution.

### 5.1.2 Test Functions and Data Sets

In comparing one ID algorithm’s effectiveness and efficiency against others over the same problem domain, instruments of validation must be applied that are standardized and recognized by the scientific community in order to be accepted. *Test functions* are widely accepted mathematical equations that evaluate a given input (a single or set of values), and return how close that input came to the test function’s defined optimal value(s), within given constraints. Competing algorithms incorporate the same test function, allowing for an objective comparison of the output in determining the superior algorithm. Our jREALGO and jMISA employ the same test functions as REALGO and MISA in comparing output, described in Section 5.2.

*Data sets* are the opposite of test functions in that they are standardized sets of data that evaluate the effectiveness and efficiency of an algorithm, based on standardized statistical measures employed. ID algorithms have two well known data sets: the MIT-DARPA Lincoln Laboratory (LL) 1999 Intrusion Detection data set [MITDARPA99] and University of California-Irvine 1999 KDD Cup data set [KDD99]. We chose the 1999 MIT-DARPA data set corpus for its large scale and variety of context-based attacks. This data set, formally introduced in Section 5.3, allows us to measure our algorithm’s performance against the LL truth set and objectively compare our results against other

algorithms applied to this same data set. Further details on the KDD Cup 99 corpus are in Appendix C.

## 5.2 C-to-Java Migration

In order to validate jREALGO and jMISA against their C-based parent programs, the Java programs are initialized to the same parameters and test functions as their C parent. For each of the 30 trials, the generation count of both implementations is equally increased to ensure correlating output between both programs. Because both programs are stochastic in nature, results vary. At the end of the 30 trials, we desire to observe Java output that is as good as or better than the C output. The sole validating factor of this experiment is effectiveness because duplicate results, more than execution time, is indicative of a proper replica.

### *REALGO* vs. *jREALGO*

In comparing REALGO to jREALGO, we discovered jREALGO is approximately 11 times less efficient but slightly more effective than REALGO. Executed 30 times between a 450 and 5000 Ab population, jREALGO appeared to maintain a proportional loss of efficiency to REALGO (Figure 36), which we compared simply for purposes of runtime observation.

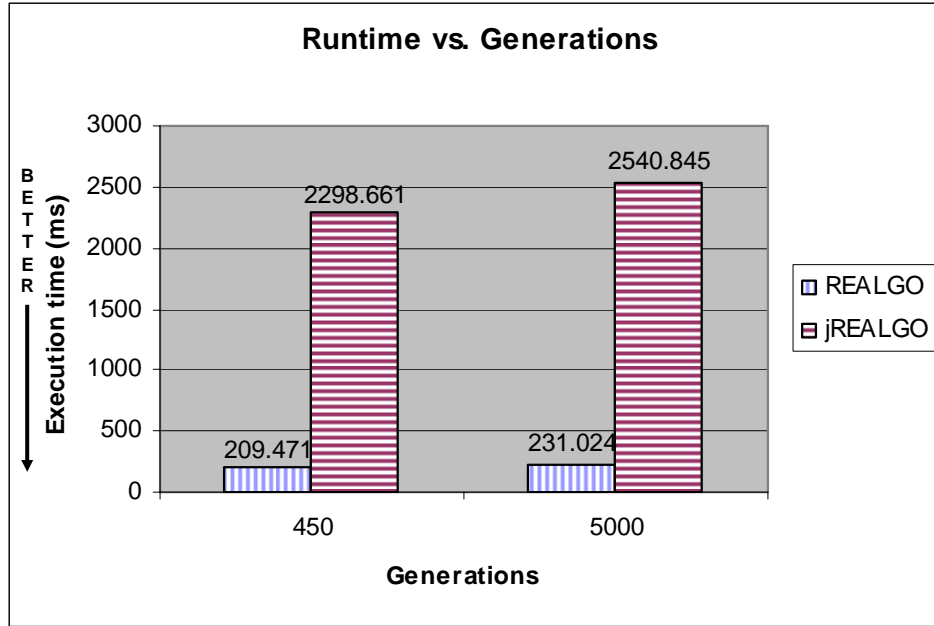


Figure 36: Runtime comparison between REALGO and jREALGO

In determining effectiveness, we chose Yao and Liu's test function (Equation 6) that was evaluated by REALGO because it was the only one in the REALGO set of experiments that yielded a non-zero optimal minimum score of -12569.5 [Yao97]:

$$\sum_{i=1}^n (-x_i \sin(\sqrt{|x_i|})) \quad (6)$$

with the landscape constrained to values ranging [-500,500].

Using Equation 6, we discovered jREALGO is slightly more effective in terms of the best and average fit antibodies for both size populations, as graphed in Figure 37 and Figure 38. This may not be true in every trial due to the stochastic nature of the algorithms and the fact they were executed in different programming languages, having differing random seed generators. In terms of standard deviation, neither algorithm is better due to

REALGO having a worse (higher) standard deviation in the smaller population but better (smaller) standard deviation in the larger population, graphed in Figure 39 and Figure 40.



Figure 37: Fitness comparison between REALGO and jREALGO: 450 generations

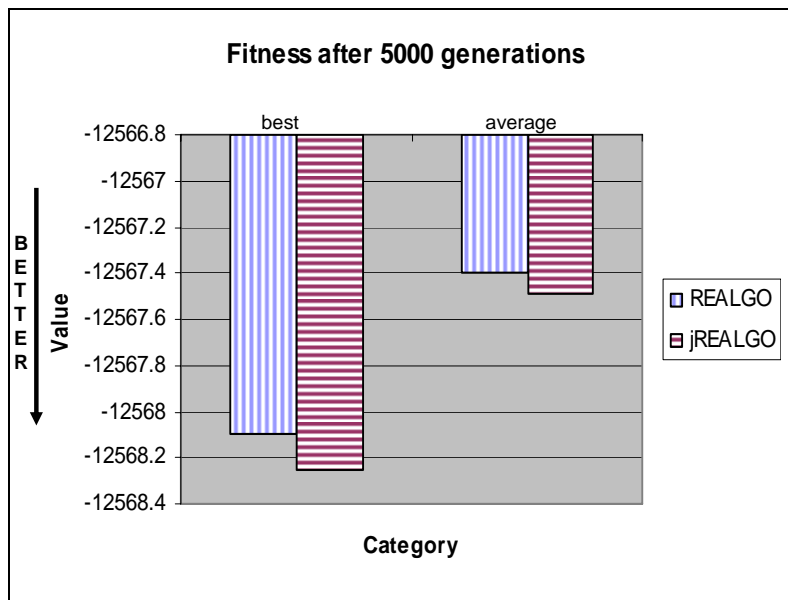


Figure 38: Fitness comparison between REALGO and jREALGO: 5000 generations

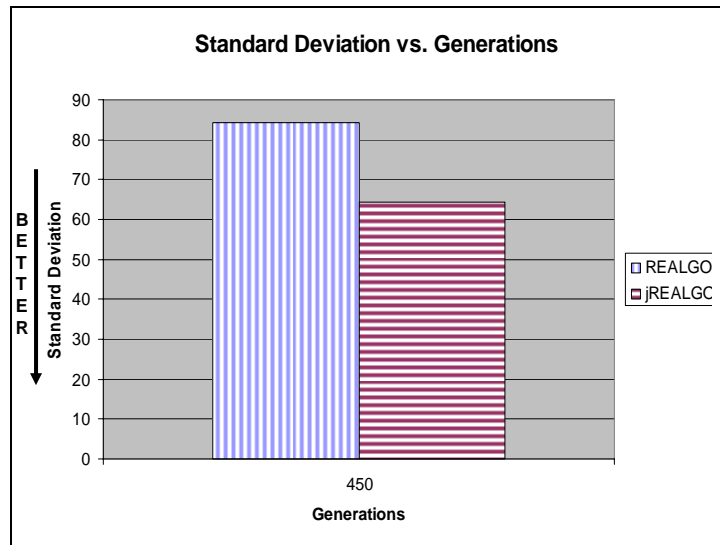


Figure 39: Standard deviation comparison between REALGO and jREALGO: 450 generations

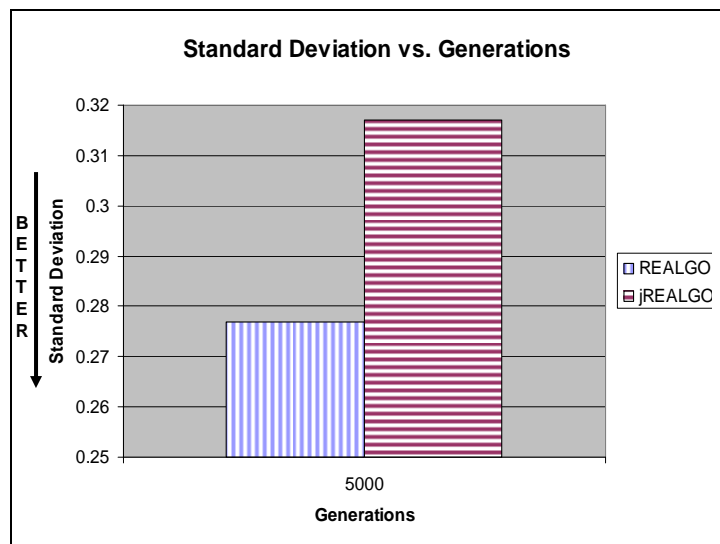


Figure 40: Standard deviation comparison between REALGO and jREALGO: 5000 generations

Therefore, due to the effectiveness of jREALGO, we conclude its migration from C to be validated.

### *MISA vs. jMISA*

In comparing MISA to jMISA, we discovered jMISA to be initially four times less efficient than MISA but that in the larger population, its factor of ineffectiveness decreased to a factor of 3.4, as graphed in Figure 41. In comparing the best, average and worst runtimes of both algorithms for both population sizes in Figure 42, we discovered the runtime deltas to be constant between best, average and worst, and that jMISA's factor of inefficiency slightly drops for the larger population.

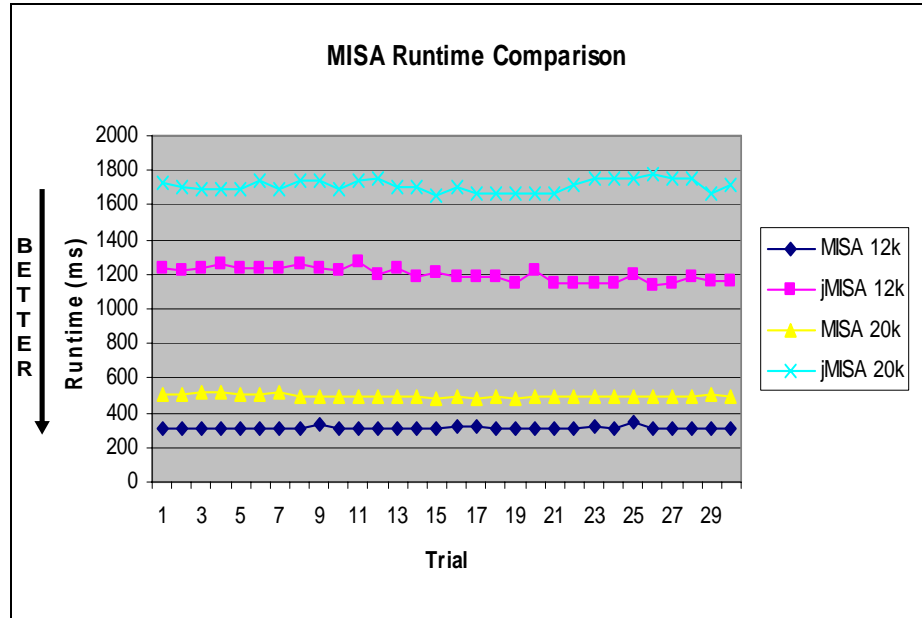


Figure 41: Runtime comparison between MISA vs. jMISA



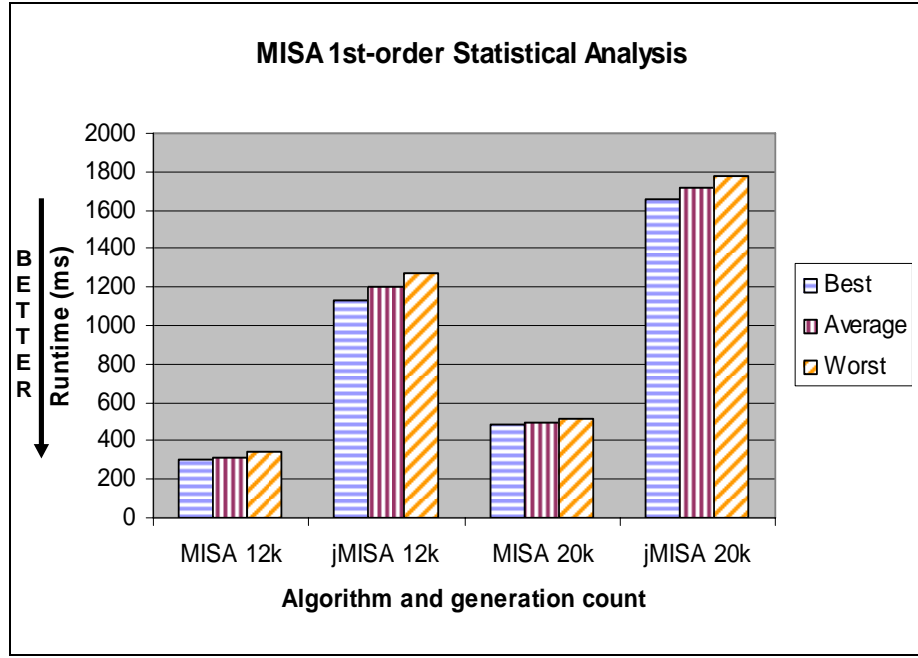


Figure 42: Statistical runtime comparison between MISA and jMISA

Test functions for MOEAs are more complex, as they require at least two variables. Hence, MISA used the Kita-proposed function [Kita96]:

$$\begin{aligned} f_1(x, y) &= -x^2 + y, \\ f_2(x, y) &= \frac{1}{2}x + y + 1 \end{aligned} \quad (7)$$

with constraints

$$x, y \geq 0, \quad 0 \geq \frac{1}{6}x + y - \frac{13}{2}, \quad 0 \geq \frac{1}{2}x + y - \frac{15}{2}, \quad 0 \geq 5x + y - 30.$$

MISA and jMISA's vector of *known Pareto Front* solutions, along with the MISA author's *true Pareto Front* value set, were input into MATLAB<sup>9</sup> which plotted the Fronts

---

<sup>9</sup> MATLAB ® a high-level language and interactive environment that enables you to perform computationally intensive tasks faster than with traditional programming languages such as C, C++, and Fortran, <http://www.mathworks.com/products/matlab/>.

depicted in Figure 43. The *genotype* space of both MISA and jMISA exhibit the same concave shape and appear to share the space of the *true Pareto Front*.

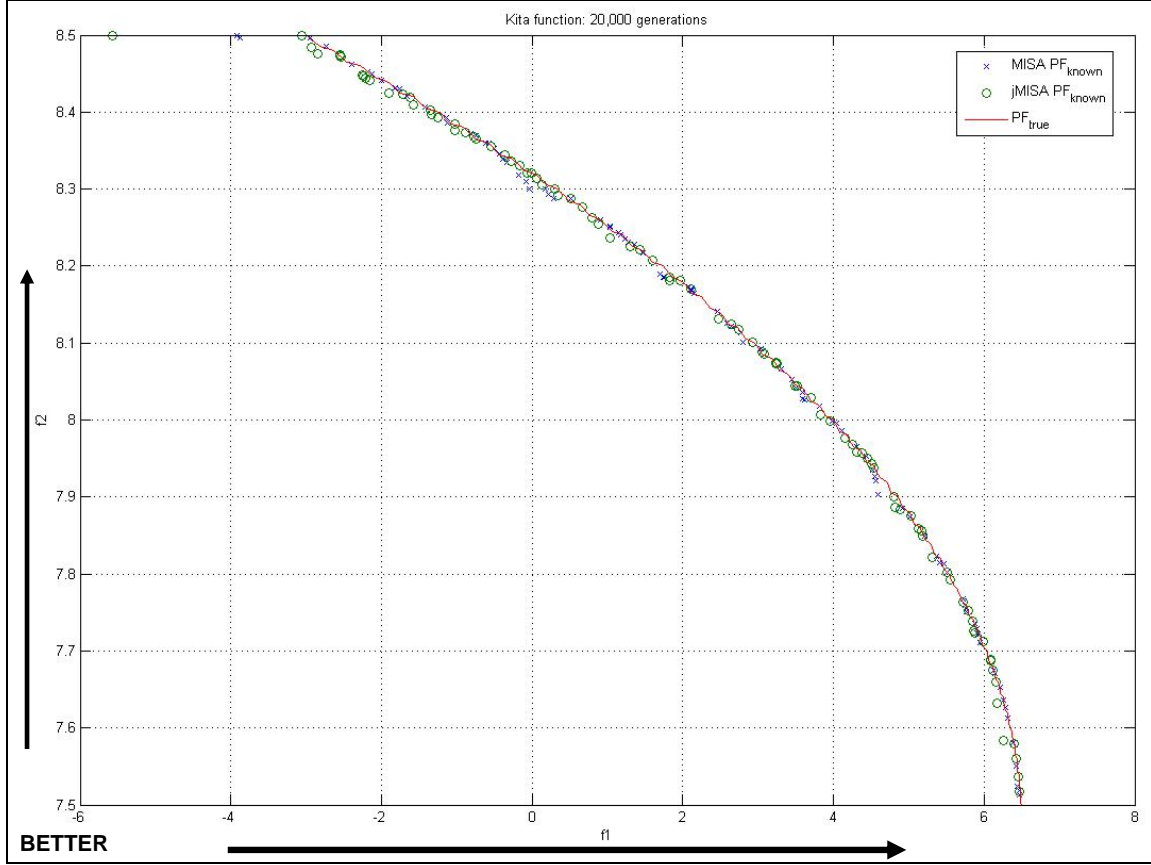


Figure 43: Plotted MISA, jMISA *known Pareto Fronts* and MISA's *true Pareto Front*

With all three data sets sorted in descending order, Euclidian distance calculation is applied to each MISA and jMISA solution point and the *true Pareto Front* point closest to it, as shown in Figure 44. Here, MISA has the preponderance of Abs with shorter distance to the *true Pareto Front* but jMISA possesses the few closest (shortest distance to the Front) points.

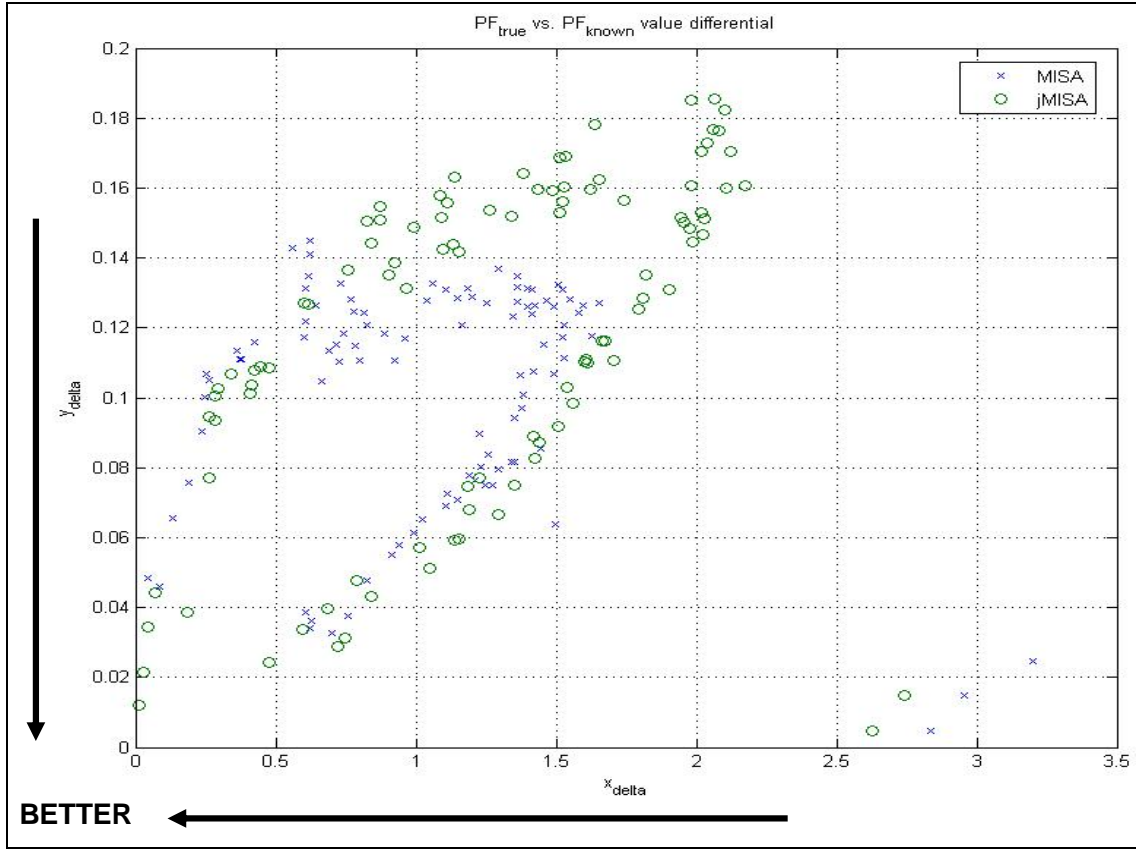


Figure 44:  $PF_{known}$  vs.  $PF_{true}$  point Euclidian-distance differential between MISA and jMISA

Based on the *genotype* similarity of both algorithms *known Pareto Front*, combined with the jMISA's shorter distance to the *true Pareto Front*, we conclude the jMISA program to be effective and validated.

### 5.3 1999 MIT-DARPA ID Data Set Evaluation

Data sets composed of simulated computer network traffic are currently the only available way of emulating a distributed computing environment containing both *self* and *non-self* events. The MIT-DARPA ID evaluation took place in both 1998 and 1999. LL coordinated with DARPA and the Air Force Research Laboratory (AFRL) to develop

several weeks’—five days per week—worth of raw TCP dump network traffic on the scale of a notional Air Force Base.

jREMISA was evaluated against the 1999 over the 1998 data set specifically because of the former’s upgrade to allow for detection of new attacks without first training on instances of the attacks [Lippmann00]. This was made possible through the inclusion of two weeks of *self*-only data, enabling jREMISA to perform *negative selection*. The intent of the *self*-only data sets is for ID systems to train against the clean sets and use that knowledge to effectively discover attack packets within the attack data sets, as explicitly recommended by LL [MITDARPA99].

Our algorithm uses the first two weeks of the 1999 corpus: the first week of *self*-only traffic to *negative-select* our Abs and the complete second week of insider-only labeled attacks to evaluate the effectiveness of our trained Abs. In evaluating jREMISA against the second week attack landscape to the fullest extent possible, we dissected as many of the IP header (context)-based labeled attacks as possible (i.e., a DoS packet sequence over a user-typed *telnet* exploit). This extraction methodology, explained in Appendix B, allowed us to procure jREMISA truth tables for 16 of the 43 LL-labeled attacks, covering all five days. When mapped to the “1999 week-two insider” landscape in Figure 45, we see attacks to be fairly distributed in both time of day and day of week, varying in size of packets, as detailed in Table 4. Figure 46 provides the trend of total event activity to *non-self* activity, for each day of the week.

In summarizing the domain breakdown and analysis of this week's data set (see Appendix B):

1. all five days of week-1's clean data sets, filtered for TCP, UDP and ICMP constitutes 7,810,861 packets (for negative selection training);
2. all five days of week-two's labeled attacks constitutes 7,275,137 packets;
3. all five day's files filtered for TCP, UDP and ICMP packets constitutes 7,199,540 packets (99.0% of the data set being jREMISA-evaluated);
4. 16 of the 43 (37.2%) attacks were successfully dissected for Chapter 5 testing;
5. of the 16 identified attacks, 53653 (0.745%) total *non-self* packets exist within the entire week's search space where 676 (1.26%) events are ICMP and the remaining 52,977 (98.7%) events are TCP.

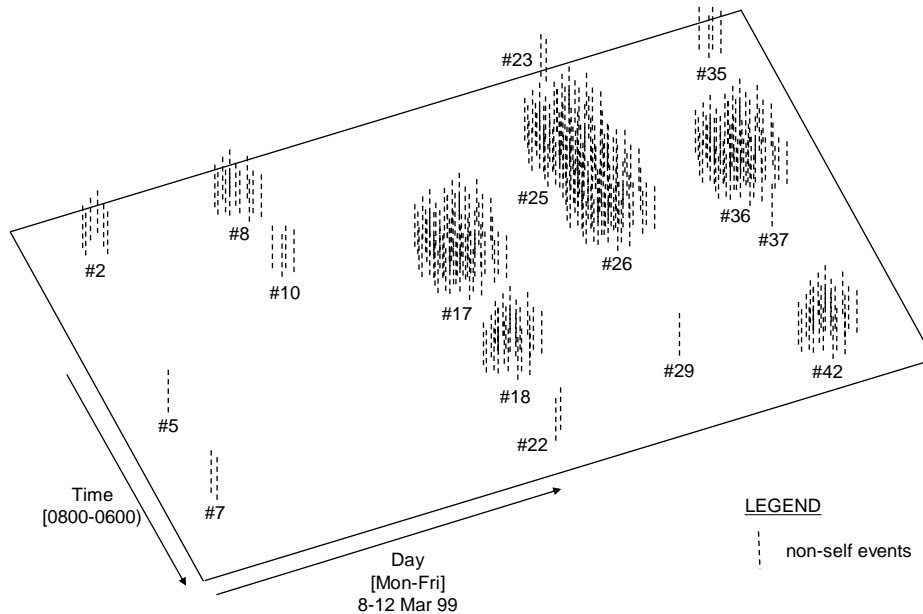


Figure 45: MIT-DARPA "1999 week-two insider" attack data set landscape with LL-labeled attacks

Attack ID	Attack Name	Protocol	Wall-Clock Time Elapsed	Number <i>Non-Self</i>
Mon, 3/8/99: 1,737,455 total events: 8 TCP, 241 ICMP (0.0143%) <i>non-self</i> events Attack day Ag ratio: 3.21% TCP, 96.79% ICMP				
2	pod	ICMP fragmented	08:50:11 – 08:50:12	241
5	land	TCP	15:57:07	1
7	ps attack	TCP – FTP	19:09:06 – 19:09:18	7
Tues, 3/9/99: 1,571,748 total events: 1552 TCP (0.0987%) <i>non-self</i> events Attack Day Ag ratio: 100% TCP				
8	portsweep	TCP [FIN]	08:44:13 – 09:11:10	1030
10	back	TCP – HTTP	10:07:30 – 10:09:30	522
Wed, 3/10/99: 995,235 total events: 15,512 TCP (1.5586%) <i>non-self</i> events Attack day Ag ratio: 100% TCP				
17	satan	TCP [SYN]	12:02:18 – 12:04:33	10504
18	mailbomb	TCP – SMTP	13:44:10 – 13:54:06	5004
22	crashiis	TCP - HTTP	23:56:00 – 23:56:06	4
Thurs, 3/11/99: 1,547,709 total events: 20,462 TCP (1.3221%) <i>non-self</i> events Attack day Ag ratio: 100% TCP				
23	crashiis	TCP – HTTP	08:04:01 – 08:04:08	4
25	portsweep	TCP	09:33:09 – 09:33:12	10056
26	neptune	TCP [SYN]	11:03:51 – 11:07:16	10401
29	land	TCP – SMTP	15:46:46	1
Fri, 3/12/99: 1,347,393 total events: 15,443 TCP, 435 ICMP (1.178%) <i>non-self</i> events Attack day Ag ratio: 97.26% TCP, 2.74% ICMP				
35	pod	ICMP fragmented	09:18:11 – 09:18:12	435
36	neptune	TCP [SYN]	11:20:10 – 11:23:35	10381
37	crashiis	TCP - HTTP	12:40:09 – 12:40:16	4
42	portsweep	TCP [SYN]	17:13:02 – 17:25:04	5058

Table 4: MIT-DARPA “1999 week-two insider” attack analysis

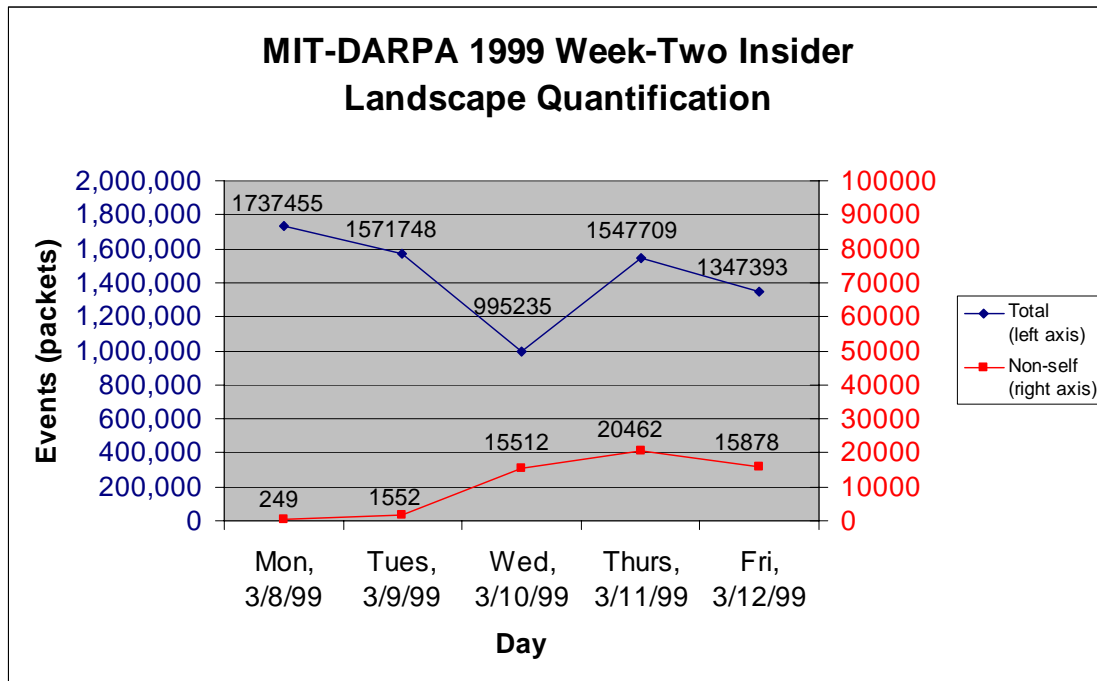


Figure 46: MIT-DARPA “1999 week-two insider” landscape quantification

All experiments performed involved all possible fields of the TCP, UDP, and ICMP headers, to fully evaluate our pattern-matching effectiveness. This means each TCP Ag was 240 bits, each UDP Ag was 170 bits and each ICMP Ag was 138 bits long.

### 5.3.1 Negative Selection Results

Our intent in testing various *negative selection* scenarios is to determine the optimal MOEA Ab population sizes and affinity threshold in order to maximize search space coverage without impinging upon *self* points in the attack-labeled evaluation. This raises the obvious question, “what is considered an optimal affinity percentage?” Our research did not find any case studies focused on determining a statistically validated percentage; hence, we start by randomly choosing equal starting population sizes and an affinity threshold and then adjust, accordingly, until we have post-execution TCP

populations similar in size to each of the Ab sets chosen by Williams in his *Warthog* experiments that evaluate this same data set: 32, 64, 128, 256, 512, 1024 and 2048 Abs. [Williams01].

Because we expect attrition and desire the largest surviving population to be at least as large as Williams' largest Ab set of 2048, we initialize all three of our populations to the next base-two power of 4096 in order to result in a post-execution TCP population size of at least 2048. We choose Friday to perform this *negative selection* gauging because of all five days of the *self-only* week, Friday represents the closest average data set size of a single day, per Table 5. In observing surviving population rates for the first time, Table 6 (graphically depicted in Figure 47) shows the range of feasible affinity threshold values until the TCP population is empty: between 37-44%.

User-required parameters for executing *negative selection* can be found in the jREMISA user manual (see Appendix D.2.2).

Day	Generations
Monday	1,477,462
Tuesday	1,222,696
Wednesday	1,710,945
Thursday	1,931,983
Friday	1,467,775

Table 5: Number of generations for each day of the 1999 week-one insider *self-only* traffic (filtered for TCP, UDP, ICMP only)



Affinity (%)	Runtime(mins)	End TCP	survived	End UDP	survived	End ICMP	survived
37	186.65	2663	65.015%	3737	91.235%	3707	90.503%
38	124.20	1563	38.159%	3372	82.324%	3513	85.767%
39	89.17	935	22.827%	2890	70.557%	3290	80.322%
40	45.27	357	8.716%	2275	55.542%	2700	65.918%
41	26.43	126	3.076%	2000	48.828%	2344	57.227%
42	16.28	34	0.830%	1431	34.937%	1997	48.755%
43	7.48	3	0.073%	808	19.727%	1259	30.737%
44	6.22	2	0.049%	618	15.088%	978	23.877%
45	4.10	0	0.000%	305	7.446%	472	11.523%
46	3.53	0	0.000%	135	3.296%	325	7.935%
47	2.90	0	0.000%	68	1.660%	184	4.492%
48	2.62	0	0.000%	29	0.708%	45	1.099%
49	2.42	0	0.000%	8	0.195%	36	0.879%
50	2.13	0	0.000%	1	0.024%	6	0.146%
51	2.13	0	0.000%	1	0.024%	5	0.122%
52	2.13	0	0.000%	0	0.000%	3	0.073%
53	2.12	0	0.000%	0	0.000%	0	0.000%

Table 6: Post-negative selection analysis of TCP, UDP, ICMP populations starting at 4096 against the Friday *self*-only data set of 1,467,775 packets

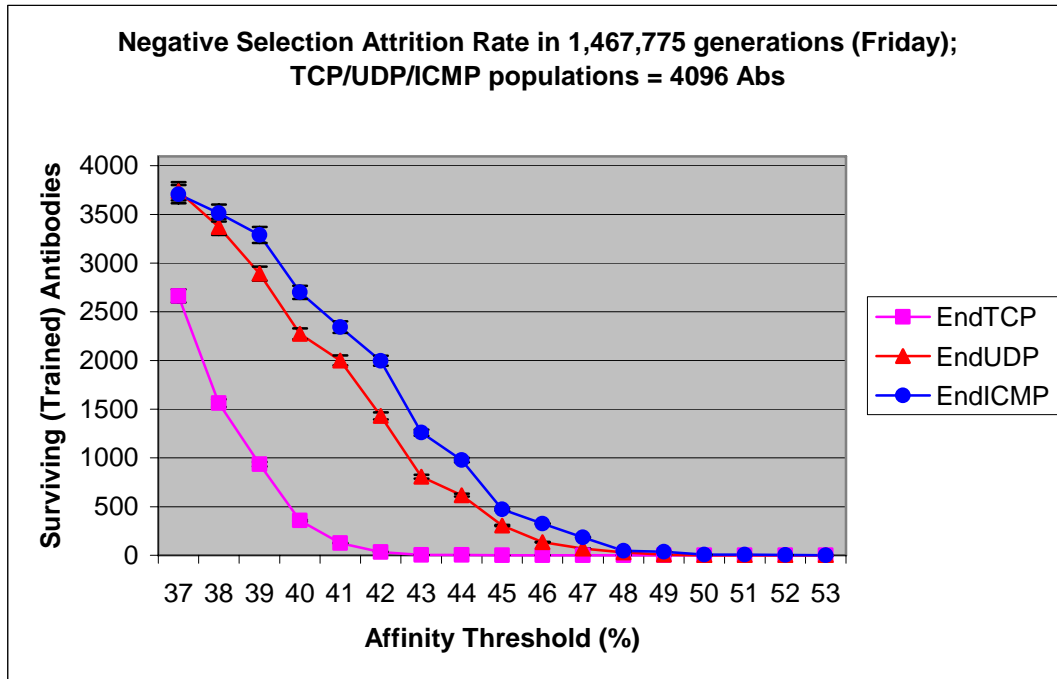


Figure 47: *Negative selection* attrition rate in 1,467,775 generations (Friday) with TCP, UDP and ICMP starting at 4,096 untrained Abs

In Figure 47, the TCP population attrits significantly quicker than the other two populations because, per Appendix B, TCP traffic quantitatively dominates the landscape. Figure 48 depicts the balance between the affinity threshold and runtime for each population starting with 4096 Abs. Conducting 30 runs for comparison and accuracy, we discovered a variance between  $\pm 4\%$  between runs, depicted by the bars within each point.

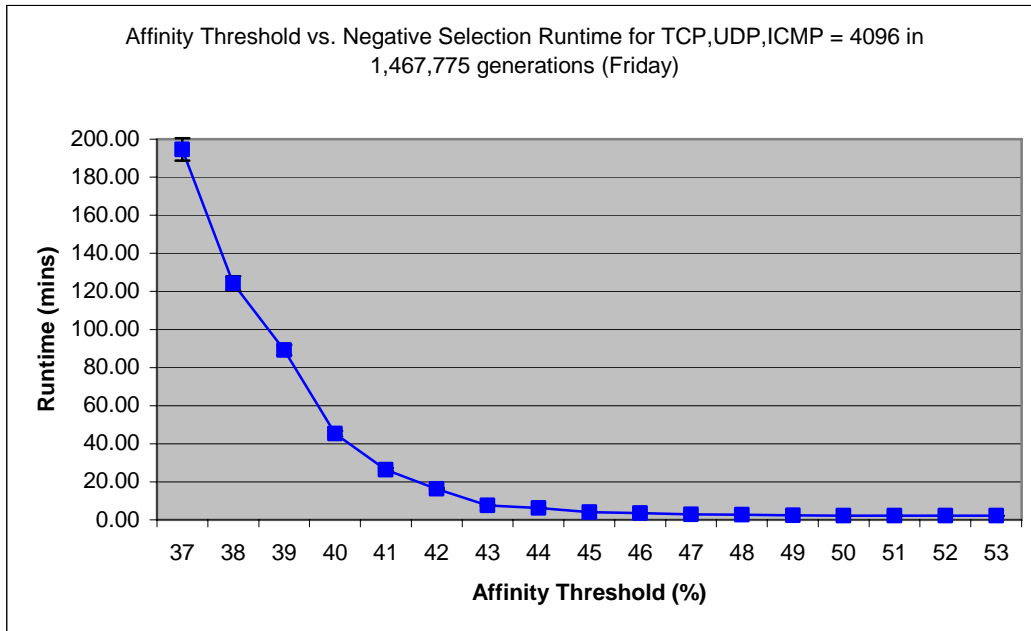


Figure 48: Affinity threshold vs. *negative selection* runtime for TCP, UDP, ICMP = 4096 untrained Abs in 1,467,775 generations (Friday)

Following *negative selection*, the trained population XML file is loaded into the core MOEA procedure of jREMISA, with several new required parameters defined (see Section D.2.3) including setting the elitism selection percentage to 5%, per [CC05].

### 5.3.2 Standalone MOEA Results

Our intent in testing the core MOEA is to determine if a protocol-segregated population manipulated by the validated strengths of REALGO and MISA can effectively detect and classify a high percentage of *self* and *non-self* traffic over the entire week's attack data set and disclose a patterned hypervolume of such effective detectors. For each day of the attack week jREMISA evaluates, it employs the *negative selection*-trained population of only that same day of the clean week, verses a trained population over the entire clean week.

While the MOEA is executing, real-time updates of the classification rates, primary and secondary population size and generation count are performed. Upon completion, output is saved to an XML file, for analysis (see Appendix D.2.3). The results of all MOEA experiment scenarios are summarized in Table 7 and Table 8. These tables provide the overarching results of executions based on the data set, affinity threshold, and *negative selection*-survived populations. Our standalone testing methodology is to perform two tests in the following order:

1. determine optimal Ab affinity threshold value based on day's false detection rate;
2. use that threshold value in performing the standalone and distributed test scenarios.

In Table 7, we perform 10 scenarios. The first six are meant to determine the affinity threshold we should choose from our feasible range to evaluate all days of the week based on the lowest false detection rate of the Thursday data set. We use the Thursday data set because it's the same data set Williams used in his *Warthog* evaluations

[Williams01]. Results of the first six scenarios conclude the lowest false detection rate when the threshold is at 39%. Hence, scenario four is compared against seven through 10, using that benchmark threshold, in comparing each day to each other. The distributed experiments performed (in Table 8) in the last three scenarios also use this benchmark.

Scenario	Day	Generations	Affinity Threshold	TCP Pop	UDP Pop	ICMP Pop	Runtime	Self Events		Non-self Events	
								True Neg%	False Neg%	True Pos%	False Pos%
1	Thurs	1547710	42%	37	86	248	39.12 m	53.78	46.22	62.6	37.4
2	“	“	41%	106	116	284	52.48 m	67.44	32.56	68.33	31.67
3	“	“	40%	315	146	341	3.61 hrs	76.10	23.90	76.92	23.08
4	“	“	39%	966	361	810	18.21hrs	85.45	14.55	97.66	2.34
5	“	“	38%	1580	423	881	2.36 days	<b>86.48</b>	13.52	92.51	7.49
6	“	“	37%	2564	462	927	5.83 days	82.52	17.48	99.71	0.29
7	Mon	1737455	39%	969	349	846	20.02 hrs	85.36	14.64	<b>99.90</b>	0.10
8	Tues	1571748	“	922	362	882	18.86 hrs	84.61	15.39	97.35	2.65
9	Wed	995235	“	920	333	798	11.69 hrs	83.37	16.63	98.26	1.74
10	Fri	1347393	“	964	376	829	13.43 hrs	83.59	16.41	96.57	3.43

Table 7: MOEA run summary: single jREMISA (highest effectiveness in bold text)

						Self Events		Non-self Events	
jREMISA machine ID	Packet range (1547709 total)	TCP Pop	UDP Pop	ICMP Pop	Runtime	True Neg%	False Neg%	True Pos%	False Pos%
Scenario 11: 2 jREMISAs, 39% affinity threshold, Thursday attack data set									
PC1	1 – 773854	966	361	810	9.44hrs	86.21	13.79	98.10	1.90
PC2	773855 – 1547709	936	344	854	9.63hrs				
Scenario 12: 3 jREMISAs, 39% affinity threshold, Thursday attack data set									
PC1	1 – 515903	966	361	810	5.09hrs	84.31	15.69	97.94	2.06
PC2	515904 – 1031807	936	344	854	6.35hrs				
PC3	1031808 – 1547709	951	357	826	6.86hrs				
Scenario 13: 4 jREMISAs, 39% affinity threshold, Thursday attack data set									
PC1	1 – 386927	966	361	810	4.33hrs	84.94	15.06	98.55	1.45
PC2	386928 – 773854	936	344	854	4.63hrs				
PC3	773855 – 1160781	951	357	826	4.86hrs				
PC4	1160782 – 1547709	954	360	822	5.09hrs				

Table 8: MOEA run summary: distributed jREMISA against Thursday data set (highest effectiveness in bold text)

Figure 49 graphically maps Table 7’s summary of jREMISA standalone classification effectiveness for each day of the attack week, with a  $\pm 1\%$  variance, as a result of multiple test runs.

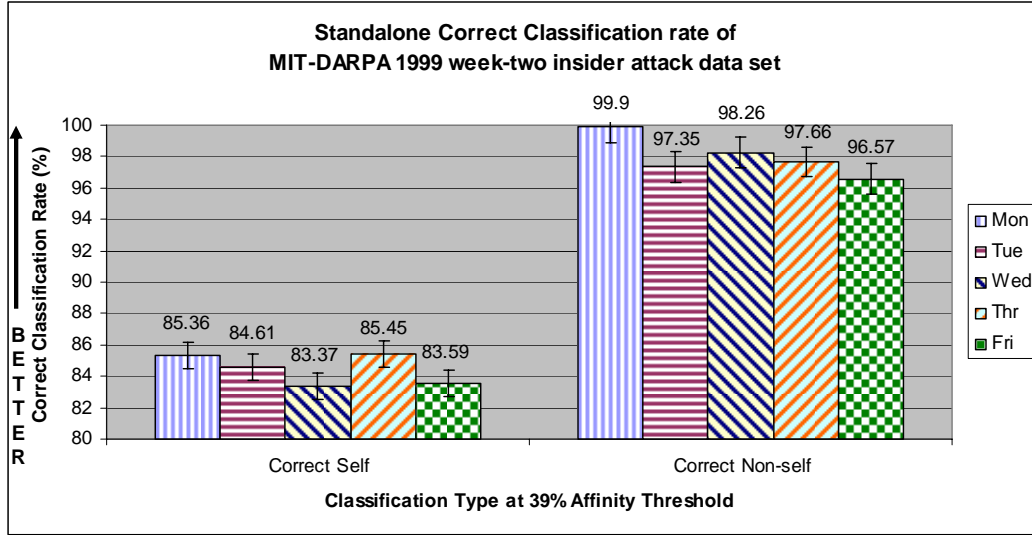


Figure 49: Standalone effectiveness against each day of the MIT-DARPA 1999 week-two insider attack data set (39% affinity threshold)

Having the overarching effectiveness results of the population, as a while, we now examine the effectiveness of the individual Ab detector. Figure 50 graphically depicts the fitness of the individual Ab detectors from the secondary tri-populations starting from a 39% affinity threshold for all Abs. Because our MOP seeks the global minimum, we desire a C-shaped boundary as close to [0,0] as possible. jREMISA maps each Abs’ correct classification fitness score and affinity threshold deviation value into an x,y-point, respectively. These two vectors are input into MATLAB for plotting<sup>10</sup>. These Pareto

<sup>10</sup> To graph secondary population Pareto Fronts into MATLAB, copy the XML file’s secondary population “Pareto-X” and “Pareto-Y” set of values into MATLAB variables  $x=[\text{Pareto-X}]$  and  $y=[\text{Pareto-Y}]$  and then type “plot(x,y,’d’)”.

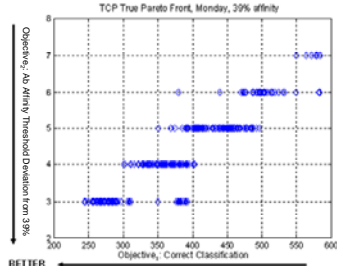
Fronts are important for the decision-maker in selecting the most optimally sized Abs with the best fitness for future ID application.

Two patterns are seen among almost all populations at the end of each day's evaluation:

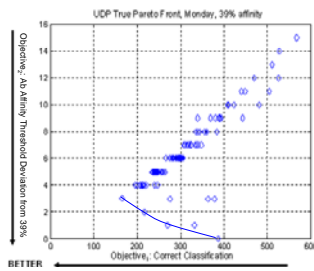
1. the Pareto Front includes Abs in +4% or +5% deviation 73% of the time;
2. Abs are concentrated at the +4% or +5% deviation value 87% of the time.

Table 7 determined our most accurate classification to be when affinity threshold was initialized at 39%. Figure 50(a-o) depicts a pattern of Pareto Fronts and Ab concentrations to have an affinity threshold between 4% and 5% higher than this initial 39% setting, leading to the conclusion of an optimally known individual Ab hypervolume between 39%-44%. Therefore, the trade-off to decision makers is picking Abs for future employment is that fitter Abs most likely will be larger, increasing the risk of future false positives while picking the more optimally sized Abs—while mitigating the false detection risk—results in a lower fitness. An exception to this is when a Pareto Front doesn't materialize, as in Figure 50(a,e) where only a single optimally known solution exists, with the pattern indicating the larger the Ab, the worse the classification fitness. In this case, decision-makers will have to decide among the next best set of Ab solution points: dominated but feasible.

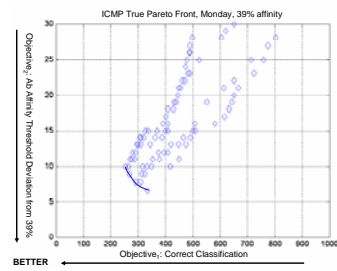
Monday, 3/8/99



(a)

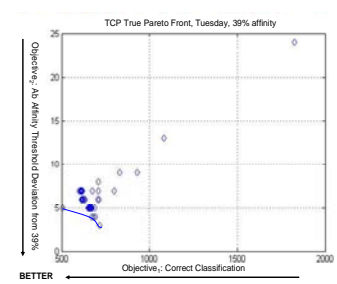


(b)

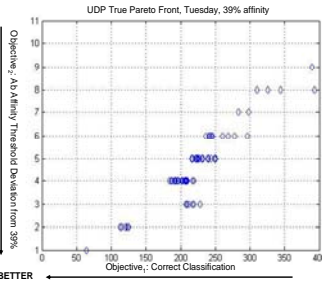


(c)

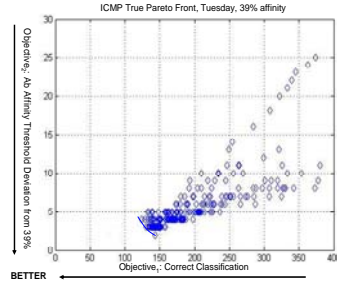
Tuesday, 3/9/99



(d)

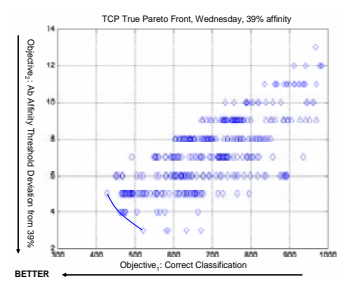


(e)

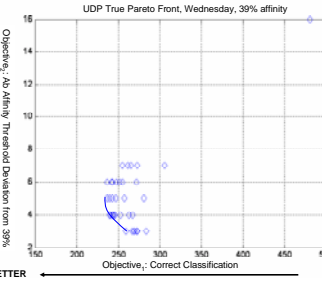


(f)

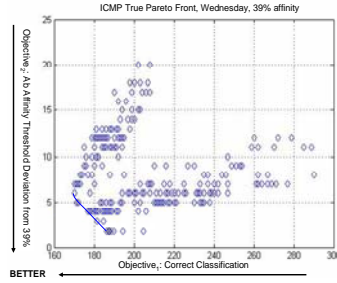
Wednesday, 3/10/99



(g)



(h)



(i)

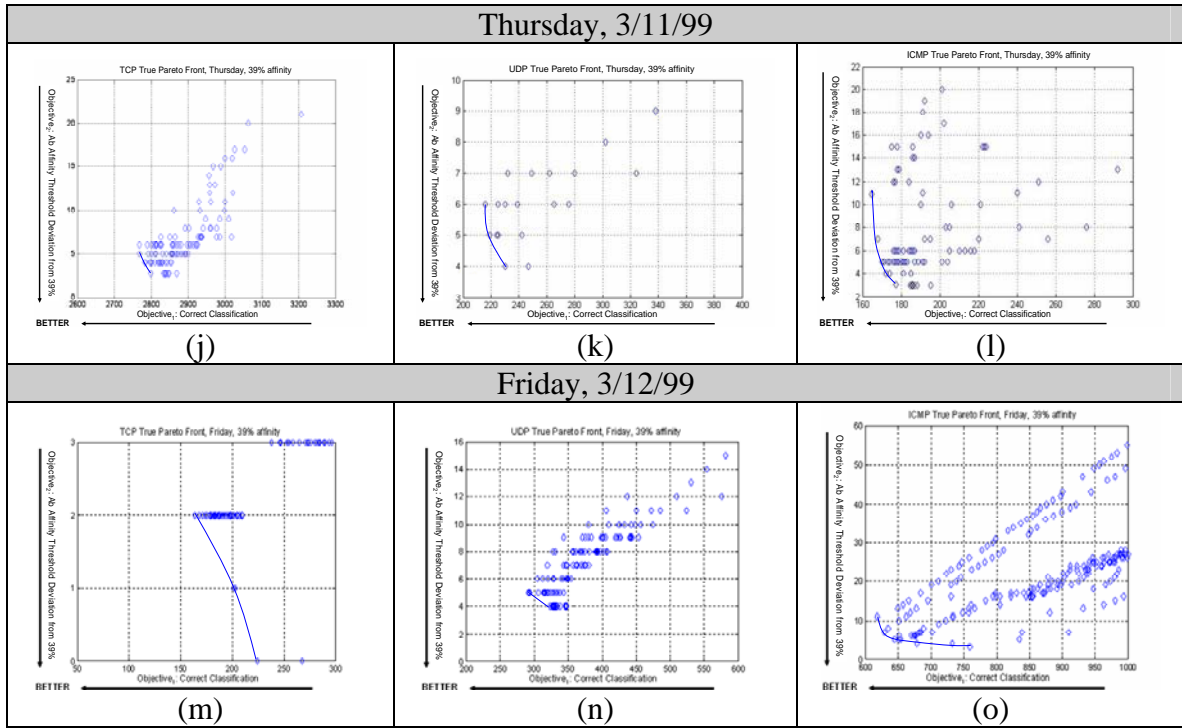


Figure 50: Post-MOEA secondary population *true Pareto Fronts*

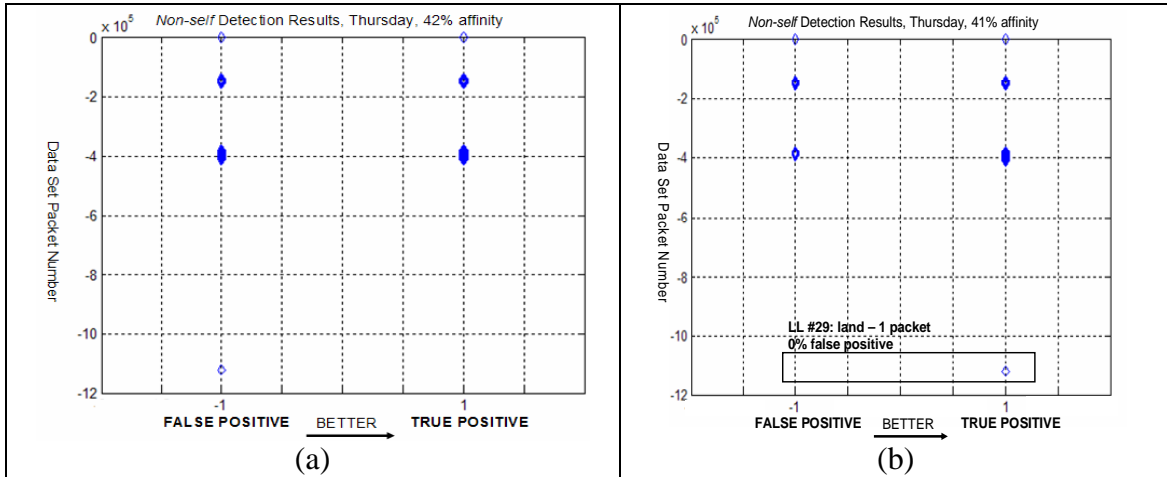
Figure 51 depicts the *attack graph*: the classification declaration of each Ab for every *non-self* data set packet of that day's data set, in order to determine the classification effectiveness of a single attack (*non-self* sequence). In plotting the *non-self* event points into MATLAB<sup>11</sup>, the x-axis denotes “-1” as false positive and “1” as true positive. The y-axis represents the packet number, increasing in a negative direction, allowing direct correlation of the classification of the attack, when referenced against the Figure 45 ID landscape.

<sup>11</sup> To graph attack results into MATLAB, copy the XML file's “X Vector” and “Y Vector” set of values into MATLAB variables  $x=[<X\ Vector>]$  and  $y=[<Y\ Vector>]$  and then type “plot(x,y,'d')”. Then scale the graph with “axis([-2 2 -<size of data set> 0])”.



When comparing Figure 51(a,b,c), we see a trend where as the Ab population's affinity threshold is linearly decreased. More “open holes” develop on the false positive side, indicating more correct classifications on the right side. Further, in Figure 51(c), two attacks—one 10401 non-contiguous packets long (LL attack ID #26<sup>12</sup>) and the other being one packet long (LL attack ID #29)—as having a 0% false positive rate for the entire population. In Figure 51(e), LL attack ID #7—seven packets long—has a 0% false positive rate; as does LL ID #22—four packets long—in Figure 51(g).

In Lippmann's discussion of the results of the off-line evaluation of the 1999 data set, he specifies that attacks were best detected when they produced a consistent signature or sequence of events in *tcpdump* data [Lippmann00]. However, the attack graphs of Figure 51(b,c) show that on two occasions, LL attack ID #29, the single-packet land attack, was detected with a 0% false positive rate, inferring jREMISA may have performed more effectively against this particular form of intrusion than the eight systems evaluated by Lippmann.



<sup>12</sup> Reference Appendix B for LL attack index mapping.

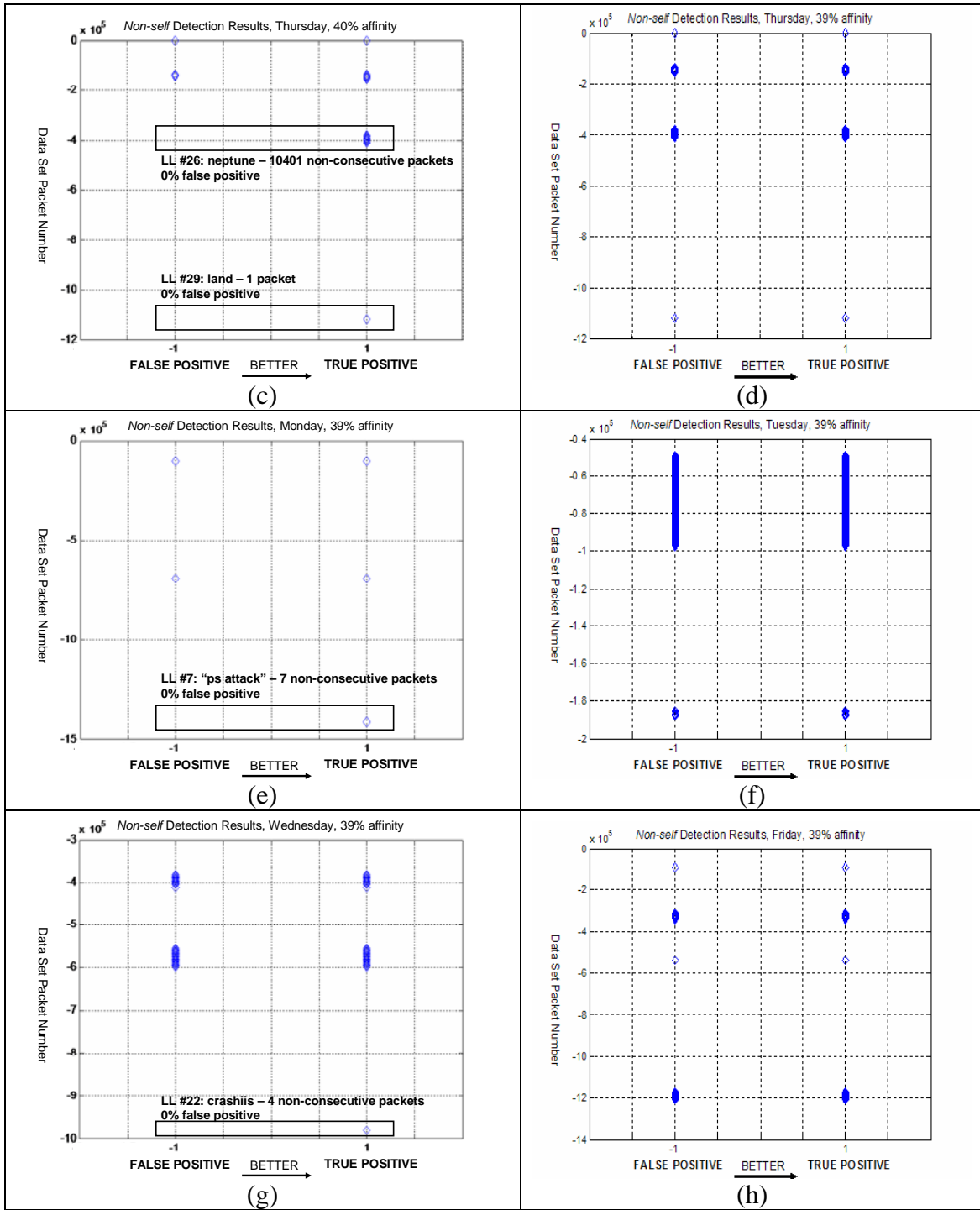


Figure 51: Post-MOEA attack graph

### 5.3.3 Distributed MOEA Results

During distributed execution and communication, we observed jREMISA broadcast, receive, and decide whether to accept received nondominated Abs into its secondary population (Figure 52). However, this methodology did not produce the pattern of synergistic effectiveness we conjectured, as the graphical mapping of Table 8 depicts in Figure 53. However, to the distributed implementation's credit, its two-jREMISA configuration achieved the highest correct *self* classification rate of all standalone and distributed tests with 86.21% at the benchmark 39% affinity threshold. In addition, it did increase evaluation efficiency almost  $n$ -fold, where  $n$  is the number of computers involved in data decomposition of the *tcpdump* file (Figure 54).

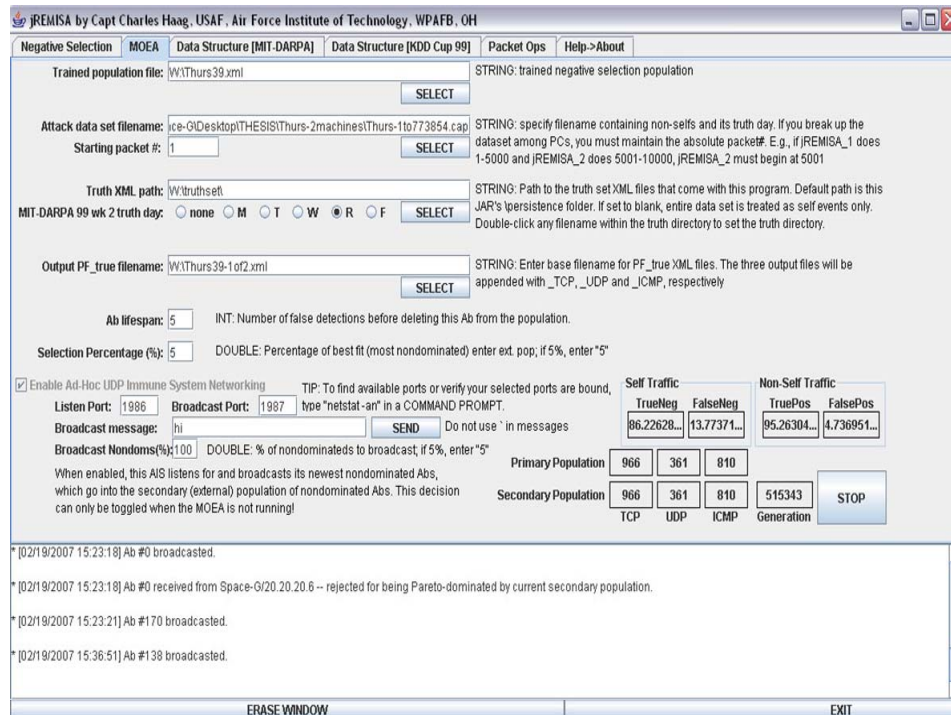


Figure 52: jREMISA screenshot of a two-system distributed island model execution

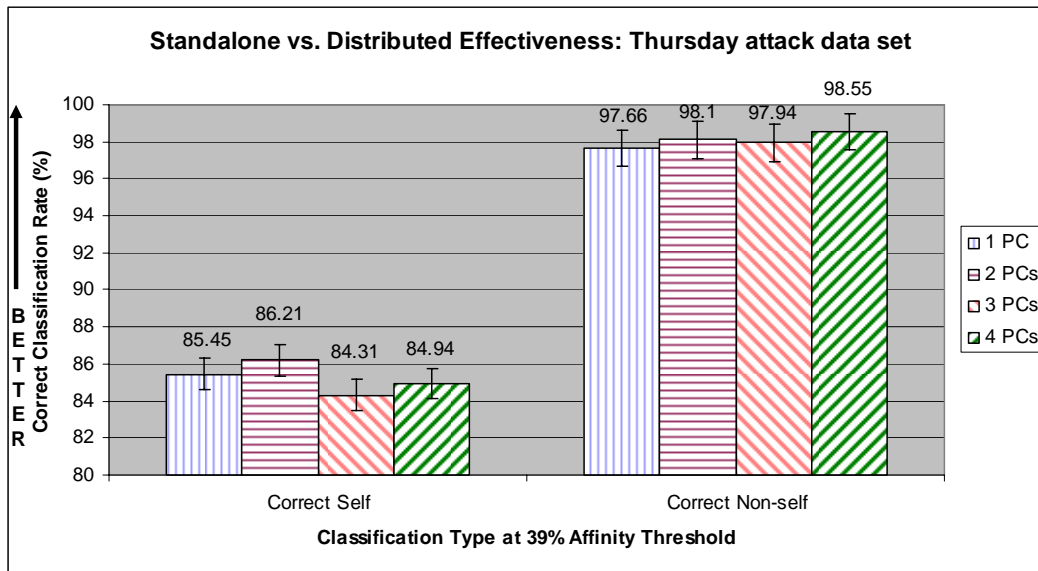


Figure 53: Standalone vs. distributed effectiveness: Thursday

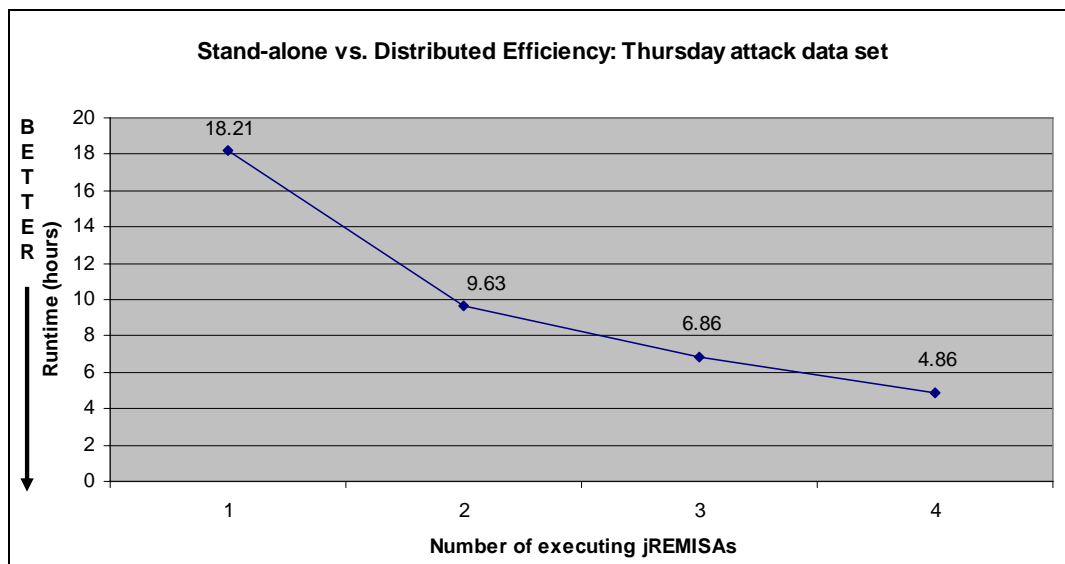


Figure 54: Data decomposition-based distributed execution: efficiency vs. number of executing jREMISAs

## 5.4 Other MIT-DARPA ID Data Set Evaluation Algorithms

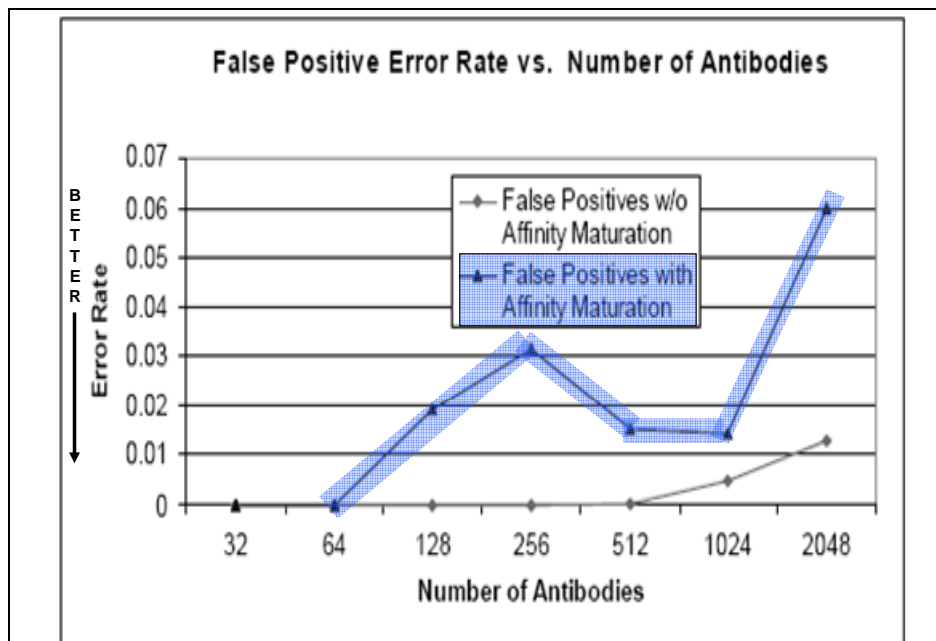
Per Garrett’s definition of “useful,” in Section 1.3, an algorithm must be distinct and effective. In this context, an algorithm is effective if it provides better or more expedient results than another in a shared benchmark test. It was difficult to compare this work to others due to our scale of evaluation, as the entire week-two’s insider data set was analyzed.

### *Williams’ Warthog*

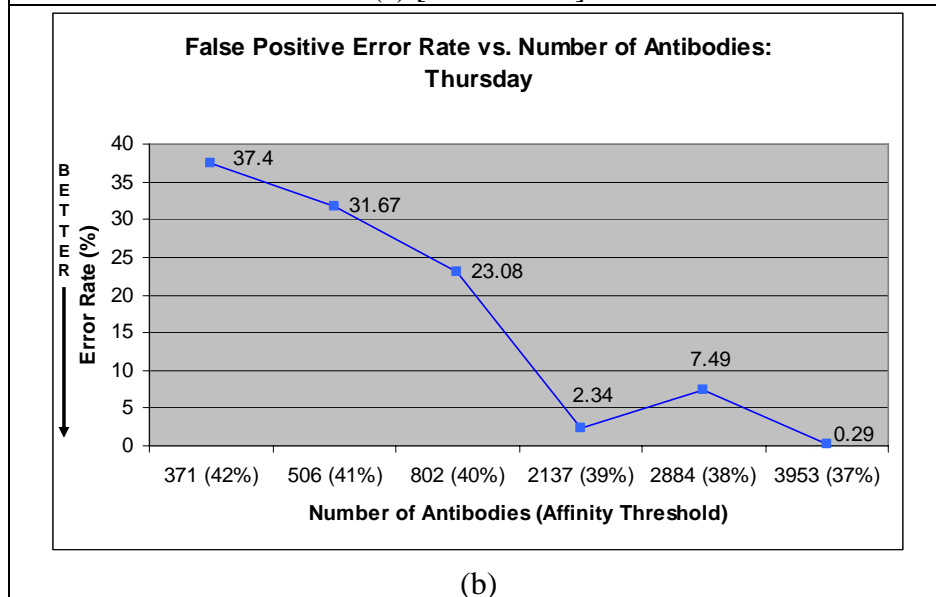
In March, 2001, Williams’ award-winning thesis focused on interactive, evolutionary search techniques, in the context of a computer immune system, to detect computer network intrusions, with particular emphasis on stealthy and *zero-day* attacks [Williams01]. His system, Warthog, trained Abs with the same LL *self*-only data sets as ours but used an attack set consisting of only 2643 LL packets from Thursday’s attack set and a set of packets generated by Nessus<sup>13</sup>. In Warthog’s “false positive error rate vs. number of antibodies” analysis, Williams concludes that as the number of Abs was increased, so did the false positive rate (Figure 55(a)). However, from jREMISA scenarios one through five, we experience a trend of continuing improvement, from a 37.4% false positive rate with 371 Abs to 0.10% with 2164 Abs (Figure 55(b)); thus giving jREMISA a better false positive rate trend.

---

<sup>13</sup> Nessus: network vulnerability scanner, <http://www.nessus.org>.



(a) [Williams01]



(b)

Figure 55: Warthog vs. jREMISA: false positive rate vs. number of antibodies

### *Other works considered*

While we discovered many algorithms applied to the 1999 MIT-DARPA data set, their experimental purpose differed from ours. The following papers were reviewed in attempting to compare the author's algorithm to ours and serve as a future avenue of research for jREMISA:

1. Gonzalez, F., Dasgupta, D., *Anomaly Detection Using Real-Valued Negative Selection*, University of Memphis, Tennessee, 2003;
2. Lydon, A., *Compilation for Intrusion Detection Systems*, Master's thesis, Ohio University, 2004;
3. Li, L., Cai, W., *Anomaly Detection using TCP header information*, George Mason University, 2004;
4. Soliman, M., El-Helw, A., *NIDS using Bloom filters*, U. of Waterloo, 2005;
5. Gaddam, S., Phoha, V., Balagani, K., *K-Means+ID3: A Novel Method for Supervised Anomaly Detection by Cascading K-Means Clustering and ID3 Decision Tree Learning Methods*, IEEE, Vol. 19, No. 3, March, 2007;
6. Shapiro, J.M., *An Evolutionary Algorithm to Generate Ellipsoid Detectors for Negative Selection*, Air Force Institute of Technology Master's Thesis, March 2005 [Shapiro05].

### **5.5 Summary**

The experiments discussed in this chapter have satisfied the established hypothesis objectives presented in Chapter 1. Section 5.2 indicates the validation of the REALGO and MISA Java replicas based on the first hypothesis objective. We were able

to build jREMISA, using these replicas as the software foundation. Section 5.3 validated the second and third hypothesis objectives of achieving “high classification rates” and discovering a discrete pattern range of optimally known Ab affinity threshold. In addition, jREMISA unexpectedly achieved 0% false classification for LL attacks 7, 22, 26 and 29, ranging between one and 10401 packets. Our fourth hypothesis objective of distributed, cooperative communication was also validated based on jREMISA behavior reported. Section 5.4 compared our results to those of another algorithm applied to the same data set day, showing jREMISA to be more effective in one area, defining the algorithm as “useful” [Garrett05].

These validated objectives conclude that jREMISA is not only a successful proof-of-concept but a *useful* ID evaluation tool in the context it provides unique features conjectured not duplicated by all other algorithms and effective in that it was shown to have better results than another algorithm against the same benchmark data set. While not yet production-grade, this software is left for continued development based on suggestions in Section 6.3. The next Chapter provides conclusions to our hypothesis objectives and suggestions for continued avenues of research.



## VI. Conclusions and Future Work

The impetus for this research stemmed from the limitations imposed by today’s predominantly-employed signature-based IDSs applied to the ID domain. Given the strengths of MOEAs and the cutting-edge research of AIS application to the ID domain, we successfully engineered a useful proof-of-concept application with a human immunological-inspired approach, utilizing evolutionary search techniques applied to the ID problem. The jREMISA MOP model of protocol-specific Ab populations computed against proven, integrated evolutionary operators from REALGO and MISA introduces a new way of accurately classifying *self* from *non-self* and responding appropriately. This chapter reflects on the conclusions drawn from previous chapters, leading to the validation of the several objectives that culminate our hypothesis.

### 6.1 Hypothesis Conclusion

This research effort set out to develop a proof-of-concept AIS-inspired MOEA applied to the ID domain. Rather than start from scratch, we wisely discovered two existing AIS algorithms—REALGO and MISA—and used them as the foundation for building jREMISA. Incorporating multiobjectivity and given a ID data set, we defined four measurable objectives based on our algorithm’s evaluation of the data set to determine if jREMISA is a useful ID domain tool. Based on the results of Chapter 5, we indicate whether our objectives are validated:

1. **VALIDATE THE MIGRATION OF EXISTING C-BASED AIS**

**ALGORITHMS INTO JAVA-BASED EQUIVALENTS.** This objective

requires duplicate output of the Java implementation that the original C code produced. Section 5.2 showed that jREALGO slightly exceeded the fitness values of REALGO against Yao and Liu’s test function. In addition, jMISA exhibits a nearly identical known Pareto Front as MISA, with jMISA possessing the solution points closest to the MISA’s *true Pareto Front*, per Euclidian distance calculations. Based on these two results, we have met this objective and can use the Java equivalents as the foundation for building jREMISA;

2. **ATTAIN THE HIGHEST CORRECT CLASSIFICATION RATE KNOWN FOR THIS PROOF-OF-CONCEPT ALGORITHM.** Section 5.3 presents jREMISA evaluation of each day of the MIT-DARPA 1999 week-two insider attack data set. The results correctly reflected *self* classification (in the 39% affinity threshold range) between 83.37% and 85.45% and *non-self* classification between 96.57% and 99.90%. In addition, all jREMISA detectors exhibited a 0% false positive rate for *non-self* event sequences that composed four attacks:
  - a. LL attack 7, “ps attack,” 7 non-consecutive packets, implying jREMISA is adept at detecting an irregular FTP session;
  - b. LL attack 22, “crashiis”, 4 non-consecutive packets, implying jREMISA is adept at detecting malformed packets for crashing Microsoft web servers;
  - c. LL attack 26, “neptune”, 10406 non-consecutive packets, implying jREMISA is adept at detecting a TCP-SYN flood DoS attack;
  - d. LL attack 29, “land”, 1 packet, on two occasions, implying jREMISA is adept at detecting a packet crafted to have the same sender and receiver.

Based on these results and the corollary that jREMISA was shown to be more effective in at least one test over another algorithm, we have met this objective's classification requirement;

3. **IDENTIFY A KNOWN OPTIMAL DETECTOR HYPERVOLUME.** Section 5.3.2's Figure 50(a-o) experimentally indicate that at least 73% of the time, Abs concentrate and form their *Pareto Front* when their hypervolume is between 39-44%, for all populations. This signifies a consistent pattern of what a desired Ab hypervolume should be in the tradeoff of its fitness score when choosing a solution Ab point for future ID domain employment. Because of these results, we have achieved this objective's requirement;

4. **VALIDATE AIS COOPERATIVE COMMUNICATION WITHIN A DISTRIBUTED ENVIRONMENT.** Section 5.3.3, Figure 52 depicts a snapshot of a jREMISA working cooperatively in evaluating the Thursday attack data set. The message console clearly shows the multiple broadcasting of newly discovered nondominated Abs and the receipt, evaluation and subsequent rejection of a broadcasted Ab from another jREMISA. As a corollary, the two-jREMISA distributed implementation had the highest correct *self* classification rate of all tests. Based on these results, we achieve this objective's cooperative communication requirement.

Because all four objectives were met, we declare our hypothesis validated and algorithm *useful*.

## 6.2 Conjectures Based on this Research

In addition to this research’s original hypothesis of validating an AIS-inspired MOEA, we submit two original conjectures and discussion of each:

1. a proposed purpose and modeling of the innate immune system;
2. the utility of IP protocol-segregated Ab populations.

### 6.2.1 Modeling the Innate Immune System

As introduced in Section 2.2, the BIS is composed of the *innate* and *adaptive immune systems*. Section 2.3 introduced the AIS as solely the computational model of the latter half of the BIS.

However, we conjecture the *innate* BIS serves a purpose for being computationally modeled. To date, AIS research applied to the ID domain has traditionally focused on packet headers (context) and not payloads (content). However, HTTP protocol payloads are desirable over other protocols for their ability to store their entire payload within one packet. For example, when logging into a server via *telnet*, each packet’s payload consists of one alphanumeric press from the keyboard. It can be quite difficult to discern a username or password for two key reasons: (i) the possibility the user may have hit the backspace key a number of times; and (ii) packets being received at arbitrary times, in an arbitrary order (assuming all packets were received). HTTP payloads, on the other hand, are not passed until the user presses the “Enter” key, passing the entire Uniform Resource Locator (URL) string into the packet. HTTP exploits are currently one of the most popular methods of vulnerability discovery and

exploitation (e.g., *phishing*<sup>14</sup>). Lippmann supports the inspection of packet payloads in his recommendations for future IDS enhancements, specifically citing the inspection of both packet contents and context [Lippmann00].

Therefore, we believe scanning HTTP packet payloads based on a database of a priori HTTP exploit strings constitutes an *innate* BIS and may be deterministically modeled as such. By integrating both the *innate* and *adaptive* BIS into one algorithm, we can complete a consistent modeling of the BIS.

### 6.2.2 Protocol-Based Antibody Populations

To date, EAs have employed a single Ab population. Per the affinity threshold, a simple distance measure between an Ab and Ag to determine *self* or *non-self* is what we term *first-order pattern matching*. ID data sets are more complex in that packets of differing protocol have a disparate number and size of payload fields. Therefore, jREMISA's initial population is a set of three protocol-specific populations: a TCP, UDP and ICMP pool. The user's selected data structure (IP, TCP, UDP and ICMP fields selected) determines the size of the search landscape and the length of the Ab. This improves pattern matching because incoming Ags are compared only to an Ab of their respective protocol, allowing "apple-to-apple" comparison of header fields. This also improves efficiency because only that subset of the entire primary population is being

---

<sup>14</sup> Defined by PC Magazine

([http://www.pcmag.com/encyclopedia\\_term/0,2542,t=phishing&i=49176,00.asp](http://www.pcmag.com/encyclopedia_term/0,2542,t=phishing&i=49176,00.asp)) as "a scam to steal valuable information such as credit card and social security numbers, user IDs and passwords." E.g., an official-looking e-mail is sent to potential victims pretending to be from their ISP, bank, or retail establishment, with the expectation some percentage of recipients respond accurately.

evaluated. Matching the Ag to the respective Ab population in this IP protocol-specific manner is what we term *second-order pattern matching*.

### 6.3 jREMISA: “The Way Ahead”

Our software, motivated by the hypotheses of this research, is developed along a two-prong approach:

1. code the foundation from existing AIS algorithms;
2. enhance with innovative, distributed and EC-inspired ideas.

All of our ideas were successfully implemented, as Chapter 5 attests. As this software was developed with the “follow-on developer” in mind, by implementing good software programming practices, we propose the following future enhancements to jREMISA:

1. implement an *innate* BIS based on known HTTP exploit strings (as explained in Section 6.2.1);
2. continue developing the evaluation capability started for the KDD Cup 99 data set (see Appendix C);
3. further develop jREMISA’s *tcpdump* decoder to facilitate protocols beyond TCP, UDP and ICMP. This recommendation is supported by Lippmann who cited that one reason many attacks were missed was due to the lack of IDS protocol support (e.g., ARP, SNMP, DNS, etc.) [Lippmann00].

Continuing to develop manageable software requires disciplined software engineering practices. As such, we suggest the motivated reader review our thoughts on software engineering principles (Appendix E) and maintaining jREMISA as open-source software (Appendix F).

It is stressed that the results of our experiments would not have been as high if we had not employed the LL truth set to guide our detectors. However, because this was a proof-of-concept algorithm against the ID domain, a guide was needed to measure reaction by the detectors. Further, our classification rates are currently far from acceptable for real-world implementation of jREMISA. For example, consider today's Air Force-level architecture comprised of 9.6 Gbps bandwidth links monitored by a team of three analysts who can realistically monitor about 240 false positives, each, before being overwhelmed, in a 24 hour period. A single 9.6 Gbps link delivers  $8.2944 \times 10^{14}$  bits of information per day. Dividing that into the maximum packet size of 16 KB yields 6,328,130,000 packets per day. Dividing the number of false positives by this number of packets results in an acceptable error rate of  $1.138 \times 10^{-7}$  [Williams07].

We highly encourage the reader to inquire about acquisition of jREMISA and its prepared data sets. Its intuitive GUI and flexibility of parameters settings allow for a combinatoric number of user-defined experiments tailored to custom search landscape size. Further, its well-commented code and implementation of multiple software design patterns allow for a minimal learning curve in altering jREMISA's programming.

#### **6.4 Continued Research Need**

On January 1, 2007, a story was published by the U.S. Department of Homeland Security (DHS) announcing a \$10.2 million dollar grant to four universities for their research into electronic terrorist activity detection [CSO07]. These universities intend to develop algorithms to find patterns and relationships in news stories and blogs utilizing mathematical tools such graph theory, dynamic data analysis, optimization, "machine

learning” and statistical analysis. This goal and methodology is synonymous to the goal of this research. Further, their intent to study information content is indicative of the need to consider network traffic payloads and supports our conjecture of modeling the *innate* BIS for HTTP payload examination. This research grant symbolizes the need for new, large-scale ID data sets and the upgrade of today’s very few and aging ID data sets.

#### **6.4.1 Suitability of the MIT-DARPA ID data sets**

This research utilized the LL-procured (1999 insider) ID data set because it currently constitutes the largest publicly available benchmark of network traffic [Mahoney03]. However, this was also the only ID data set evaluated because of the general lack of public domain data sets; a consequence of proprietary data privacy concerns combined with the difficulty level in accurately simulating Internet traffic [Mahoney03].

To make matters worse, the LL data sets have been criticized from many angles as an overall inaccurate ID domain model. McHugh, in his assessment of the complete LL corpus, declares the following fallacies [McHugh00]:

1. questionable traffic collection methodology;
2. attack taxonomy;
3. lack of statistical evidence validating real-world Air Force network traffic;
4. low traffic rates;
5. relative uniform distribution of the four major attack categories;
6. skewed distribution of the victim hosts;
7. overall flat network topology.



To evaluate these claimed simulation artifacts, Mahoney and Chan developed a simplistic anomaly detection system they claimed “could not possibly work” [Mahoney03]. Their system was trained on the first week and evaluated against the second week of network traffic—the same as jREMISA. Their results indicated a 45% attack detection rate (79 of 177 attacks), with 43 false alarms, making them competitive with the top systems involved in the original evaluation.

These results appear to give merit to McHugh’s claims of the LL corpus, in which he does not accompany answers with the many questions he raised [Mahoney03]. Mahoney’s recommendation to accurately modeling an ID data set is to simply add real traffic to the simulation. Not withstanding privacy concerns, the benefits of a real-traffic data set include [Mahoney03]:

1. eliminating the need to simulate traffic and label attacks;
2. factoring in the IP protocols introduced since 1999;
3. a greater volume of encrypted traffic, allowing for a more accurate modeling of today’s network traffic composition.

Mahoney and Chan researched other ID data sets, such as *Internet Traffic Archive*<sup>15</sup>, but found it unsuitable for its lack of application payload. They conclude by suggesting the need for a new benchmark and, because of the proliferation of application-payload encrypted traffic, migrate anomaly detection systems from NIDS, “on-the-wire,”

---

<sup>15</sup> The *Internet Traffic Archive* is a moderated repository to support widespread access to traces of Internet network traffic, <http://ita.ee.lbl.gov>.

to HIDS, which have the ability to evaluate the decrypted payload, after its delivery (see Section 2.1.1).

#### **6.4.2 “Cyber Storm”: the next ID data set?**

In February of 2006, InfoWorld.com reported that the DHS had just completed, “the first full-scale government-led cyber attack simulation” [InfoWorld06]. A public report of the results and lessons learned was to be released mid-2006, said Andy Purdy, acting director of the DHS National Cyber Security Division. DHS called this simulation a “sophisticated cyber attack, involving 115 organizations in the U.S., Canada, the U.K., Australia and New Zealand,” in addition to private companies such as Microsoft, VeriSign Inc. and Symantec Corp. Participating governmental agencies included the DoD, Department of Justice, the U.S. State Department and the National Security Agency. In February of 2007, InfoWorld.com reported that the DHS is planning “Cyber Storm 2” to be conducted in March of 2008 [InfoWorld07]. It’s billed to include a greater number of participants than the first, particularly with respect to number of international participants.

In consideration of the aging ID data sets of today, this author conjectures the opportunity may exist to become involved in this exercise to determine its potential value as the next real-world ID data set.

### **6.5 Summary**

This Chapter began with a review of the objectives needed to be met in order to validate our hypothesis and the Chapter 5 experiments that met the hypothesis’

objectives. Following, we contribute ideas to unexplored areas of the AIS field. We conclude with the future for jREMISA and its continued research need for helping solve the ID problem.

Overall, this research effort validated our hypothesis that an AIS-inspired MOEA, composed of segregated populations and proven EA operators of past AIS algorithms, is useful and effective against the ID problem domain. Furthermore, we believe this software to be the first AIS to apply multiobjectivity to the ID domain; specifically, the MIT-DARPA data set. It is our hope that this proof-of-concept software be further investigated, with the possibility it may bring us yet closer to solving the ID problem.

## Bibliography

- [Anderson80] Anderson, J., *Computer Security Threat Monitoring and Surveillance*, James P. Anderson Co., Fort Washington, PA, 1980.
- [Bäck96] Bäck, T., *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New York, 1996.
- [BDNG06] Balachandran, S., Dasgupta, D., Nino, F., Garrett, D., *A General Framework for Evolving Multi-Shaped Detectors in Negative Selection*, Computer Science Department, University of Memphis, Tennessee, 2006, World Wide Web URL: <http://ais.cs.memphis.edu/papers/ais/2006/Multishaped-Detectors.pdf>.
- [Burnet50] Burnet, F.M., *Clonal selection and after*, Theoretical Immunology, Bell, G.I., Perelson, A.S., Pimgley Jr. (Eds.), P.J., Marcel Dekker Inc., 1978, pp.63-85.
- [Busetti03] Busetti, F., *Simulated Annealing Overview*, World Wide Web URL [www.geocities.com/francorbusetti/saweb.pdf](http://www.geocities.com/francorbusetti/saweb.pdf).
- [Castro05] De Castro, L.N., *Forum: Adaptation, Bio-Inspiration, Complexity*, PowerPoint presentation, International Symposium on Bio-Inspired Computing, Johor, Malaysia, 5-7 September, 2005, World Wide Web URL: [http://bic05.fsksm.utm.my/files/Forum\\_Leandro.ppt](http://bic05.fsksm.utm.my/files/Forum_Leandro.ppt).
- [CC05] Coello, C., Cortés, N., *Solving Multiobjective Optimization Problems Using an Artificial Immune System*, Genetic Programming and Evolvable Machines, Vol. 6, pp.163-190, 2005.
- [CDIS01] Williams, P., Anchor, K., Bebo, J., Gunsch, G., Lamont, G., *CDIS: Towards a Computer Immune System for Detecting Network Intrusions*, Graduate School of Engineering and Management, Air Force Institute of Technology, 2001.
- [Chen04] Chen, T., *Intrusion Detection for Viruses and Worms*, International Engineering Consortium, November, 2004, World Wide Web URL: <http://engr.smu.edu/~tchen/papers/iec2004.pdf>.
- [Chertoff01] Assistant Attorney General Michael Chertoff's Testimony before the House Subcommittee on Crime, world Wide Web URL: [http://www.usdoj.gov/criminal/cybercrime/cybercrime61201\\_MChertoff.htm](http://www.usdoj.gov/criminal/cybercrime/cybercrime61201_MChertoff.htm).

- [CSEP07] Computational Science and Engineering Program Website, World Wide Web URL: <http://csep.hpcc.nectec.or.th/>.
- [CSO07] Daniel, D., U.S. Department of Homeland Security, *Research Aims to Detect Online Terrorist Activity*, World Wide Web URL: [http://www.csoonline.com/read/010107/brf\\_terror\\_id\\_pf.html](http://www.csoonline.com/read/010107/brf_terror_id_pf.html).
- [Cusumano95] Cusumano, M., Selby, R., *Microsoft Secrets: How the World's Most Powerful Software Company Creates Technology, Shapes Markets, and Manages People*, Simon & Schuster, New York, 1995.
- [CVL02] Coello, C., Van Veldhuizen, D., Lamont, G.B., *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, New York, 2002.
- [Darwin64] Darwin, C., *On the Origin of Species, 1st Edition* (facsimile: 1964), Harvard University Press, Cambridge, MA, 1859.
- [Dasgupta99] Dasgupta, D., *Artificial Immune Systems and their Applications*, Springer publishing, 1999.
- [DBP91] De Boer, R.J., Perelson, A.S., *Size and Connectivity as Emergent Properties of a Developing Immune Network*, Journal of Theoretical Biology, volume 149, pp.381-424, 1991.
- [DCVZ99a] De Castro, L.N., Von Zuben, F., *Artificial Immune Systems: Part I – Basic Theory and Applications*, Technical Report TR-DCA 01/99, December, 1999.
- [DCVZ99b] De Castro, L.N., Von Zuben, F., *Artificial Immune Systems: Part II – A Survey of Applications*, Technical Report DCA-RT 02/00, February, 2000.
- [DeBoer92] De Boer, R.J., Segel, L.A., Perelson, A.S., *Pattern Formation in One- and Two-dimensional Shape-Space Models of the Immune System*, Journal of Theoretical Biology, volume 155, pp.295-333, 1992.
- [Detours96] Detours, V., Sulzer, B., Perelson, A.S., *Size and Connectivity of the Idiotypic Network are Independent of the Discreteness of the Affinity Distribution*, Journal of Theoretical Biology, volume 183, pp.409-416, 1996.
- [DMC91] Dorigo, M., Maniezzo, V., Colorni, A., *The ant system: an autocatalytic optimizing process*, Technical Report TR91-016, Politecnico Di Milano, 1991.

- [DPST06] J.. Dréo, A. Pétrowski, P. Siarry, and E. Taillard, *Metaheuristics for Hard Optimization: Methods and Case Studies*, Springer-Verlag, Berlin, Germany, 2006.
- [Dreher95] Dreher, H., *The Immune Power Personality: 7 Traits You Can Develop to Stay Healthy*, Plume Books, 1996.
- [EC1] Bäck, T., Fogel, D.B., Michalewicz, T., *Evolutionary Computation I: Basic Algorithms and Operators*, Institute of Physics (IoP) publishing, 2002.
- [ELR06] Edge, K., Lamont, G., Raines, R., *A Retrovirus Inspired Algorithm for Virus Detection & Optimization*, GECCO '06, July 8-12, 2006.
- [Farmer86] Farmer, J., Packard, N., Perelson, A., Physica, D. , *The Immune system, adaptation, and machine learning*, Volume 2, Issue 1-3, Oct-Nov, 1986, pp. 187-204.
- [FF04] Freeman, Eric. ,Freeman, Elizabeth, Sierra, K., Bates, B., *Head First Design Patterns*, O'Reilly Media, Inc., 2004.
- [Forrest95] Dasgupta, D., Forrest, S., *Tool breakage detection in milling operations using a negative-selection algorithm*, Technical Report No. CS95-5, Department of Computer Science, University of New Mexico, 1995.
- [Garrett05] Garrett, S., *How Do We Evaluate Artificial Immune Systems?*, Evolutionary Computation, Volume 13, Issue 2, June, 2005.
- [GGKK03] Grama, A., Gupta, A., Karypis, G., Kumar, V., *Introduction to Parallel Computing, Second Edition*, Addison-Wesley, 2003.
- [GG99] George, A.J.T., Gray, D., *Receptor editing during affinity maturation: taking leaps through the landscape*, from *Immunology Today*, volume 20, pp. 196.
- [Greensmith03] Greensmith, J., *New Frontiers for an Artificial Immune System*, Digital Media Systems Laboratory, HP Laboratories Bristol, HPL-2003-204, 7 October 2003, World Wide Web URL: <http://www.hpl.hp.com/techreports/2003/HPL-2003-204.pdf#search=%22%22New%20Frontiers%20for%20an%20artificial%20immune%20system%22%22>.
- [Halloran05] Halloran, T., Lt. Col, USAF, *GRASP: Designing Objects with Responsibilities – Lesson 17*, Microsoft PowerPoint presentation, CSCE593: Introduction to Software Engineering, Department of Electrical and Computer Engineering, Graduate School of Engineering and Management, Air Force Institute of Technology, Ohio, USA, 2005.

- [HF00] Hofmeyr, S., Forrest, S., *Architecture for an Artificial Immune System*, IEEE Transactions on Evolutionary Computation, 2000.
- [Hightower95] Hightower, R.R., Forrest, S., Perelson, A.S., *The Evolution of Emergent Organization in Immune System Gene Libraries*, in L. J. Eschelman (Eds.), Proceedings of the Sixth International Conference on Genetic Algorithms, Morgan Kaufmann, San Francisco, California, pp.344-350, 1995.
- [Hightower96] Hightower, R.R., Forrest, S., Perelson, A.S., *The Baldwin Effect in the Immune System: Learning by Somatic Hypermutation*, in R.K. Belew and M. Mitchell (Eds.), Adaptive Individuals in Evolving Populations, Addison-Wesley, MA, pp.159-167, 1995.
- [Hofmeyr00] Hofmeyr, S.A., *An Overview of the Immune System, Tutorial about computational immunology*, World Wide Web URL: <http://www.cs.unm.edu/~immsec/html-imm/immune-system.html>.
- [HS02] Halloran, T.J., Scherlis, W., *High Quality and Open Source Software Practices (Position Paper)*, Second Workshop on Open Source Software Engineering, International Conference on Software Engineering (ICSE) 2002, World Wide Web URL: <http://opensource.ucc.ie/icse2002/HalloranScherlis.pdf>.
- [HSE03] Halloran, T.J., Scherlis, W., Erenkrantz, J., *Beyond Code: Content Management and the Open Source Development Portal (Position Paper)*, Workshop on Open Source Software Engineering, ICSE, 2003, World Wide Web URL: <http://www.fluid.cs.cmu.edu:8080/Fluid/fluid-publications/HalloranScherlis.pdf>.
- [HTD97] Hertz, A., Taillard, E., Dominique de Werra, *Tabu Search: Local Search in Combinatorial Optimization*, John Wiley & Sons Ltd., 1997.
- [HWGL02] Harmer, P., Williams, P., Gunsch, G., Lamont, G.B., *An Artificial Immune System Architecture for Computer Security Applications*, IEEE Transactions on Evolutionary Computation, Vol. 6, No. 3, June 2002.
- [InfoWorld06] Gross, G., IDG News Service, *U.S. DHS completes large-scale cyber exercise*, InfoWorld.com, World Wide Web URL: [http://www.infoworld.com/article/06/02/10/75280\\_HNdhs\\_1.html](http://www.infoworld.com/article/06/02/10/75280_HNdhs_1.html)
- [InfoWorld07] McMillan, R., IDG News Service, *US government readying massive cybersecurity test*, InfoWorld.com, World Wide Web URL: [http://www.infoworld.com/article/07/02/12/HNcyberstorm2\\_1.html](http://www.infoworld.com/article/07/02/12/HNcyberstorm2_1.html)

- [Janeway97] Janeway Jr, C. A. and Travers, P., *Immunobiology: The Immune System in Health and Disease*, Artes Médicas (Portuguese), 2d edition, 1997.
- [Java04] Singer, M., *Java Enterprise Gains Broader Support*, September, 2004, World Wide Web URL: <http://www.internetnews.com/ent-news/article.php/3402341>.
- [KDD99] Hettich, S. and Bay, S. D. (1999). The UCI KDD Archive. Irvine, CA: University of California, Department of Information and Computer Science., World Wide Web URL: <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>.
- [Kita96] Kita, H., Yabumoto, Y., Mori, N., Nishikawa, Y., *Multi-objective optimization by means of the thermodynamical genetic algorithm*, Parallel Problem Solving from Nature—PPSN IV, Voight, H.-M., Ebeling, W., Rechenberg, I., Schwefel, H.-P. (Eds.), Lecture Notes in Computer Science, Springer-Verleg: Berlen, Germany, pp. 504-512, September, 1996.
- [Lamont06] Lamont, G.B., *Multi-Objective Evolutionary Algorithms: What, Why, and Where? A Tutorial (revision 2)*, Class handout, CSCE 886, Evolutionary Computation, Department of Electrical and Computer Engineering, Graduate School of Engineering and Management, Air Force Institute of Technology, Ohio, USA, 2006.
- [LamontACO06] Lamont, G.B., *Swarm intelligence: the origins of ant colony optimization techniques*, Class handout, CSCE 886, Evolutionary Computation, Department of Electrical and Computer Engineering, Graduate School of Engineering and Management, Air Force Institute of Technology, Ohio, USA, 2006.
- [Lippmann00] Lippmann, R., Haines, J., Fried, D., Korba, J., Das, K., *The 1999 DARPA Off-Line Intrusion Detection Evaluation*, Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, MA, 2000
- [LJ07] LinuxJournal.com: Understanding IDS for Linux, World Wide Web URL: <http://www.linuxjournal.com/node/5616/print>.
- [Mahoney03] Mahoney, M., Chan, P., *An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection*, Technical Report CS-2003-02, Computer Science Department, Florida Institute of Technology, 2003.
- [Marmelstein99] Marmelstein, R., Van Veldhuizen, D., Harmer, P., Lamont, G., *Modeling & Analysis of Computer Immune System using Evolutionary Algorithms*, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, December 1999.



- [McGee07] McGee, P., *Building Better Antibody Therapeutics*, Drug Discovery & Development, World Wide Web URL: <http://www.dddmag.com/ShowPR.aspx?PUBCODE=090&ACCT=1600000100&ISSUE=0701&RELTYPE=DEV&PRODCODE=00000000&PRODLETT=AG&CommoCount=0>.
- [McHugh00] McHugh, J., *Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory*, Association for Computing Machinery (ACM) Transactions on Information and System Security (TISSEC), Vol. 3, Issue 4, 262-294, 2000.
- [MFH02] Mockus, A., Fielding, R., Herbsleb, J., *Two Case Studies of Open Source Software Development: Apache and Mozilla*, ACM Transactions on Software Engineering and Methodology, Vol. 11, No. 3, pp. 309-346, July, 2002.
- [MGL04] Maniezzo, V., Gambardella, L.M., De Luigi, F., *Ant Colony Optimization*, New Optimization Techniques in Engineering, by G.C. Onwubolu and B.V. Babu, pp. 101-117, Springer-Verlag, Berlin, Heidelberg, 2004.
- [Michalewicz04] Michalewicz, Z., Fogel, D., *How to Solve It: Modern Heuristics, Second Edition*, Springer-Verlag, Berlin, Germany, 2004.
- [Middlemiss06] Middlemiss, M., *Positive and Negative Selection in a Multilayer Artificial Immune System*, The Information Science Discussion Paper Series, No. 2006/03, University of Otago, January, 2006.
- [MITDARPA99] MIT Lincoln Laboratory – DARPA Intrusion Detection Evaluation, World Wide Web URL: <http://www.ll.mit.edu/IST/ideval/>.
- [Nath07] Nath, A., *Evolutionary Algorithms in a Nutshell*, Portable Document Format (PDF) slideshow, Association for Computing Machinery, University of Illinois Student Chapter, September, 2006.
- [Navy80] Denning, D., *An Intrusion Detection Model*, IEEE Transactions on Software Engineering, Vol. 13, pp.222-232, February, 1987.
- [NIST01] Bace, R., Mell, P., *Intrusion Detection Systems, NIST Special Publication on Intrusion Detection Systems*, National Institute of Standards and Technology, 2001.
- [NSS07] The NSS Group – Europe’s foremost independent network and security testing organisation: Gigabit Intrusion Detection Systems (IDS), World Wide Web URL: [http://www.nss.co.uk/WhitePapers/gigabit\\_ids.htm#Host%20IPS%20\(HIPS\)](http://www.nss.co.uk/WhitePapers/gigabit_ids.htm#Host%20IPS%20(HIPS)).

- [Osyczka85] Osyczka, A., *Multicriteria optimization for engineering design*. In Gero, J.S., editor, *Design Optimization*, pp.193-227, Academic Press, 1985.
- [Perelson96] Perelson, A.S., Hightower, R., Forrest, S., *Evolution and Somatic Learning of the V-Region Genes*, Research in Immunology, volume 147, pp.202-208, 1996.
- [PO79] Perelson, A.S., Oster, G.F., *Theoretical studies of clonal selection: minimal antibody repertoire size and reliability of self-non-self discrimination*, Journal of Theoretical Biology, volume 81, no. 4, pp.645-70, 1979.
- [Porter06] Porter, B., *Approaching Zero: A Study in Zero-Day Exploits – Origins, Cases, and Trends*, Norwich University Journal of Information Assurance (NUJIA), Vol. 2, Issue 2, June, 2006.
- [Raymond00] Raymond, E.S., *The Cathedral and the Bazaar*, Thyrsus Enterprises, 2000, World Wide Web URL: <http://www.tuxedo.org/~esr/>.
- [Rensberger96] Rensberger, B., *In Self-Defense*, from “Life itself,” Oxford University Press, p.212-228.
- [S&C92a] Seiden, P.E., Celada, F., *A Model for Simulating Cognate Recognition and Response in the Immune System*, Journal of Theoretical Biology, volume 158, pp.329-357, 1992.
- [S&C92b] Seiden, P.E., Celada, F., *A Computer Model of Cellular Interactions in the Immune System*, Immunology Today, volume 13(2), pp.56-62, 1992.
- [S&P88] Segel, L., Perelson, A.S., *Computations in Shape Space: A New Approach to Immune Network Theory*, in *Theoretical Immunology, Part Two*, (Ed.) A.S. Perelson, pp.321-343.
- [SF07] SecurityFocus.com: Intrusion Detection Terminology (Part Two), World Wide Web URL: <http://www.securityfocus.com/infocus/1733>.
- [SFP93] Smith, R.E., Forrest, S., Perelson, A.S., *Population Diversity in an immune system model: Implications for genetic search*, Foundations of Genetic Algorithms, Whitley (Ed.), L.D., Morgan Kaufmann Publishers: San Mateo, CA, Vol. 2, pp.153-165, 1993.
- [Shapiro05] Shapiro, J.M., *An Evolutionary Algorithm to Generate Ellipsoid Detectors for Negative Selection*, Air Force Institute of Technology Master’s Thesis, March 2005.

- [Smith97] Smith, D.J., Forrest, S., Hightower, R.R., Perelson, S.A., *Deriving Shape Space Parameters from Immunological Data*, Journal of Theoretical Biology, volume 189, pp.141-150, 1997.
- [Starlab] World Wide Web URL: <http://users.pandora.be/richard.wheeler1/ais/inn.html>.
- [Stevens94] Stevens, R.W., *TCP/IP Illustrated, Volume 1: The Protocols*, Addison-Wesley, 1994.
- [Symantec04] Symantec Internet Security Threat Report, Trends for January 1, 2004 – June 30, 2004, Volume VI, September, 2004, World Wide Web URL: [http://eval.veritas.com/mktginfo/enterprise/white\\_papers/ent-whitepaper\\_symantec\\_internet\\_security\\_threat\\_report\\_vi.pdf](http://eval.veritas.com/mktginfo/enterprise/white_papers/ent-whitepaper_symantec_internet_security_threat_report_vi.pdf).
- [Symantec06] Symantec Internet Security Threat Report, Trends for January – June, 2006, Volume X, September 2006, World Wide Web URL: [http://www.symantec.com/specprog/threatreport/ent-whitepaper\\_symantec\\_internet\\_security\\_threat\\_report\\_x\\_09\\_2006.en-us.pdf](http://www.symantec.com/specprog/threatreport/ent-whitepaper_symantec_internet_security_threat_report_x_09_2006.en-us.pdf).
- [Timmis02] De Castro, L.N., Timmis, J., *Artificial Immune Systems: A New Computational Intelligence Approach*, Springer publishing, 2002.
- [Timmis04] Timmis, J., *Artificial Immune Systems: An Overview*, PowerPoint presentation, Computing Laboratory, University of Kent at Canterbury, UK, March, 2004.
- [Warthog01] Williams, P., Anchor, K., Bebo, J., Gunsch, G., Lamont, G.B., *Warthog: Towards a Computer Immune System for Detecting “Low and Slow” Information System Attacks*, White Paper, Graduate School of Engineering and Management, Air Force Institute of Technology, March, 2001.
- [Williams01] Williams, P., *WARTHOG: Towards a Computer Immune System for Detecting “Low and Slow” Information System Attacks*, Air Force Institute of Technology Master’s Thesis, March 2001.
- [Williams07] Williams, P., “Government-acceptable false positive error rate.” Electronic message, 28 Feb 07.
- [Worm07] Computer worm – Wikipedia, the free encyclopedia, January, 2007, World Wide Web URL: [http://en.wikipedia.org/wiki/Computer\\_worm](http://en.wikipedia.org/wiki/Computer_worm).

[WS07] WindowSecurity.com: Intrusion Detection Systems (IDS) Part 2, World Wide  
Web URL: <http://www.windowsecurity.com/articles/IDS-Part2-Classification-methods-techniques.html>.

[Yao97] Yao, X., Liu, Y., *Fast Evolution Strategies, Control & Cybernetics*, Vol. 26, No. 3, pp. 467-496, 1997.

## Appendix A: ID-Domain Stochastic Search Algorithms

This appendix further elaborates on Section 2.4.2's introduction to types of stochastic search algorithms employed in the ID domain. While not utilized in our research because of their inability to conform to jREMISA's data structure or algorithm, it is worth explaining what applications they serve for the purpose of choosing the most appropriate one to apply to a particular problem domain. Seven popular stochastic algorithms discussed are:

1. simulated annealing (SA);
2. tabu search (TS);
3. genetic algorithm (GA);
4. evolutionary strategy (ES);
5. evolutionary programming (EP);
6. ant colony optimization (ACO).

### A.1 Simulated Annealing (SA)

Computational SA was developed in 1983 to deal with highly nonlinear and combinatorial optimization problems [Buseti03] and to escape local optima [Michalewicz04]. The algorithm itself was inspired by the metallurgical *annealing* technique where a controlled heating and cooling process of a material through a temperature  $T$  is intended to produce a uniform distribution of crystals with the lowest possible internal energy. A controlled  $T$  decrease leads to a uniform, crystallized solid (stable) state, corresponding to an absolute minimum of energy [DPST06]. Conversely,

the rapid lowering of  $T$  results in *quenching*—an amorphous structure (metastable) state leading to local minimums of energy.

Computationally speaking, SA is a strategy to approach a landscape's globally optimum solution with given constraints, traversing its various sub-optimal solutions within a *neighborhood*, beginning at the highest  $T$ . This  $T$  allows for a search over the largest possible area in a stochastic *random walk* manner. If our initial  $T$  and *cooling rate* are optimally chosen, we restrict the number of sub-optimal choices made, stabilizing our system and shrinking our neighborhood. As the neighborhood becomes small where  $T$  finally tends to zero, our algorithm degenerates from stochastic to deterministic because only improvements are accepted and, thus, a DFS completes the greedy search for the optimum solution. However, if  $T$  is rapidly decreased (analogous to quenching), we most likely end up in a (amorphous) local minimum.

As mentioned above, in metallurgy, a quenched material results in an amorphous (defective) structure. However, this defect can be overcome by re-heating and cooling, again. Synonymously, SAs major advantage over other methods is the ability to escape local minima by increasing  $T$ . Consider the analogy of a bouncing ball that can bounce over mountains, from valley to valley (local minima). At highest (initial)  $T$ , the ball can bounce to any valley (Random Walk). As it does so,  $T$  decreases, resulting in less bounce. If  $T$  is low enough where the ball cannot bounce out of a valley,  $T$  can be increased, giving enough bounce for the ball to escape the valley. Algorithm 7 outlines the generic SA algorithm.

---

```

1  procedure simulatedAnnealing
2  begin
3     $t \leftarrow 0$ 
4    initialize  $T$ 
5    select a current point  $v_{\text{current}}$  at random
6    evaluate  $v_{\text{current}}$ 
7    repeat
8      repeat
9        select a new point  $v_{\text{new}}$  in the neighborhood of  $v_{\text{current}}$ 
10       if  $\text{eval}(v_{\text{current}}) < \text{eval}(v_{\text{new}})$ 
11         then  $v_{\text{current}} \leftarrow v_{\text{new}}$ 
12       else if  $\text{random}[0,1) < e^{\frac{\text{eval}(v_{\text{new}}) - \text{eval}(v_{\text{current}})}{T}}$ 
13         then  $v_{\text{current}} \leftarrow v_{\text{new}}$ 
14       until (termination-condition)
15        $T \leftarrow g(T, t)$ 
16        $t \leftarrow t + 1$ 
17     until (halting-criterion)
18  end

```

---

Algorithm 7: Basic simulated annealing algorithm [MICHALEWICZ04]

In order to execute SA, the following information is required a priori and problem domain-specific [Michalewicz04]:

1. what defines a solution?
2. what is the neighborhood makeup of a solution?
3. what is the cost of a solution?
4. how is the initial solution determined?
5. how is the initial temperature  $T$  determined?
6. how is the cooling ratio  $g(T, t)$  determined?
7. what defines the termination condition?
8. what defines the halting criterion?

## A.2 Tabu Search (TS)

Tabu Search, in its purest form, is deterministic. Is it not until the *tabu list*, for remembering last values visited in order not to re-visit them, is included, that it becomes stochastic—the implementation method more commonly used. The roots of Tabu Search date back to the 1970s. The algorithm, itself, eventually became a refined SA procedure in which the introduction of a *tabu* list in memory maintains visited points, forcing the search process to explore only unvisited areas [HTD97]. TS has now become an established approximation technique which has been validated to beat many classical procedures. TS is essentially deterministic, as opposed to SA, but can be modified to include probabilistic elements, making it more nondeterministic [Michalewicz04]. For example, as with SA's *T* regulator, TS can escape local optima through probabilistic control of an *aspiration level*. Algorithm 8 outlines the generic TS algorithm.

---

```
1  procedure tabuSearch
2  begin
3    select a current point  $v_{\text{current}}$  at random
4    evaluate  $v_{\text{current}}$ 
5     $\vec{v}_{\text{tabu}} \leftarrow v_{\text{current}}$ 
6    Repeat
7      evaluate every point in the neighborhood of  $v_{\text{current}}$ 
8      select a new point  $v_{\text{new}}$  in the neighborhood of  $v_{\text{current}}$ 
9      if ( $\text{eval}(v_{\text{current}}) > \text{eval}(v_{\text{new}})$ )  $\wedge v_{\text{new}} \notin \vec{v}_{\text{tabu}}$  then
10          $v_{\text{current}} \leftarrow v_{\text{new}}$ 
11          $\vec{v}_{\text{tabu}} \leftarrow \vec{v}_{\text{tabu}} \cup v_{\text{current}}$ 
12      $t = t + 1$ 
13 until (halting-criterion)
14 end
```

---

Algorithm 8: Basic tabu search algorithm [MICHALEWICZ04]



As TS is modeled after SA, some of the same a priori information is required and problem domain-specific [Michalewicz04]:

1. what defines a solution?
2. what is the neighborhood makeup of a solution?
3. what is the cost of a solution?
4. how is the initial solution determined?
5. what defines the halting criterion?

### **A.3 Genetic Algorithm (GA)**

A GA is a search technique used in computing to find true or approximate solutions to optimization and search problems based on Darwin's "survival of the fittest" theory of evolution [Darwin64]. In natural evolution, each species searches for beneficial adaptations in an ever-changing environment (domain). As species evolve, these new attributes are encoded into the chromosomes of the individual members. While information does change via random mutation, the real force behind evolutionary development is the exchange of the best chromosomal building blocks between two chromosomes, during breeding [CSEP07]. GAs differ from traditional optimizations in four respects. They:

1. manipulate the encoding of the variables vice the variables, themselves;
2. search from one population to another, verses individual to individual;
3. use objective function information, not derivatives;
4. use probabilistic vice deterministic transition rules.

The pseudocode for a standard GA is described in Algorithm 9.

---

```

1  procedure GA
2  begin
3     $g := 0$ ; /* generational counter */
4    initialize P(g)
5    evaluate P(g) /* compute fitness values */
6    while ( $\iota(P(g)) \neq \text{true}$ ) do
7       $g := g + 1$ ;
8      select: P(g) from P(g-1)
9      crossover: P'(g)
10     mutate: P'(g)
11     evaluate: P'(g)
12     P := survivors(P,P',g)
13 od
14 end

```

---

Algorithm 9: Genetic Algorithm pseudocode

#### A.4 Evolutionary Strategy (ES)

ES are similar to GAs in that they, too, simulate evolution. The difference arises in their origin of application. While GAs were designed to solve discrete or integer optimization problems, ES were first applied to continuous parameter optimization problems associated with laboratory experiments (e.g., they use real-vector coding representation) [CSEP07]. Recombination involves taking the mean of each element (allele) of the parent vector. Because jREMISA employed a bit string data structure, ES did not conform to our work. ES and GAs are just two algorithms in the EAs collective.

#### A.5 Evolutionary Programming (EP)

EP is similar to the GA idea but its data structure is not restricted to the chromosome [Nath07]. Solutions can have any data structure with various mutation methodologies possible based on the particular solution. Similar to jREMISA,

recombination tends not to play a role. As jREMISA was restricted to a fixed chromosome, the EP did not meet our requirements.

## **A.6 Ant Colony Optimization (ACO)**

ACO is a paradigm for designing metaheuristic algorithms to solve combinatorial optimization problems based on the collective foraging behavior of ants. The first ACO algorithm was introduced in 1991 [DMC91] with the essential trait being the combination of a priori information about the structure of a promising solution with a posteriori information about the structure of previously obtained good solutions [MGL04]. ACOs drive a low-level, constructive solution in a population framework that randomizes the construction in a *Monte Carlo*<sup>16</sup> way. A Monte Carlo combination of different solutions elements is also suggested by GAs but in the case of ACOs, the probability distribution is explicitly defined by the previously obtained solutions. Initial ACO applications included [LamontACO06]:

1. the traveling salesman problem (TSP);
2. quadratic assignment problem;
3. graph colouring;
4. job-shop scheduling;
5. sequential ordering;
6. vehicle routing.

---

<sup>16</sup> *Monte Carlo* methods involve simulations dealing with stochastic events; they employ a purely random search where any selected trial solution is fully independent of any previous choice and its outcome. The current “best” solution and associated decision variables are stored as a comparator [CVL02].

In the real world, ants initially wander randomly and upon finding food, return to the colony while laying down pheromone that temporarily enables trail remembrance. Hence, if other ants find such a path, they likely will follow it vice continue wandering randomly. Over time, however, the pheromone trail evaporates, reducing its attractive strength. However, as more ants traverse this path, the more pheromone is laid, providing a stronger attraction to that particular path. This evaporation process has the advantage of avoiding convergence to a locally optimal solution. Thus, when one ant finds a good (pheromone-strong) path from the colony to the food source (objective), other ants are more likely to follow that path, eventually resulting in a single, optimal path.

ACOs have the advantage over SA and GA when the problem domain graph (e.g., TSP) changes dynamically. When this happens, the ant colony can be run continuously and adapt to changes in real-time. Figure 56 depicts an example of ants finding more food ( $F$ ) below the barrier than above it, resulting in a more optimal pheromone trail developing below the barrier than above.

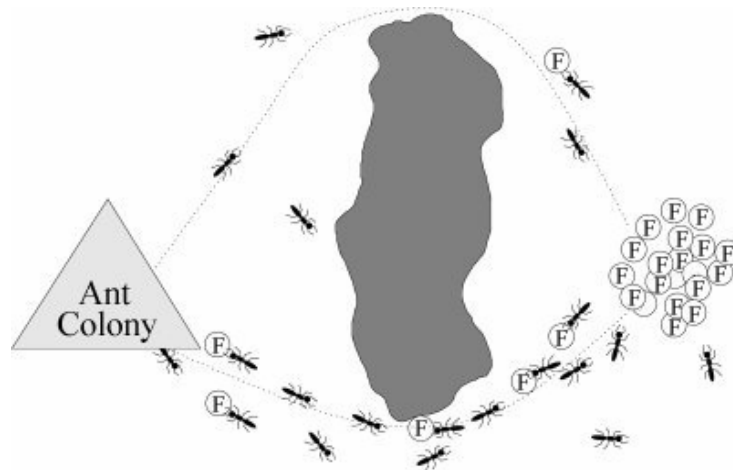
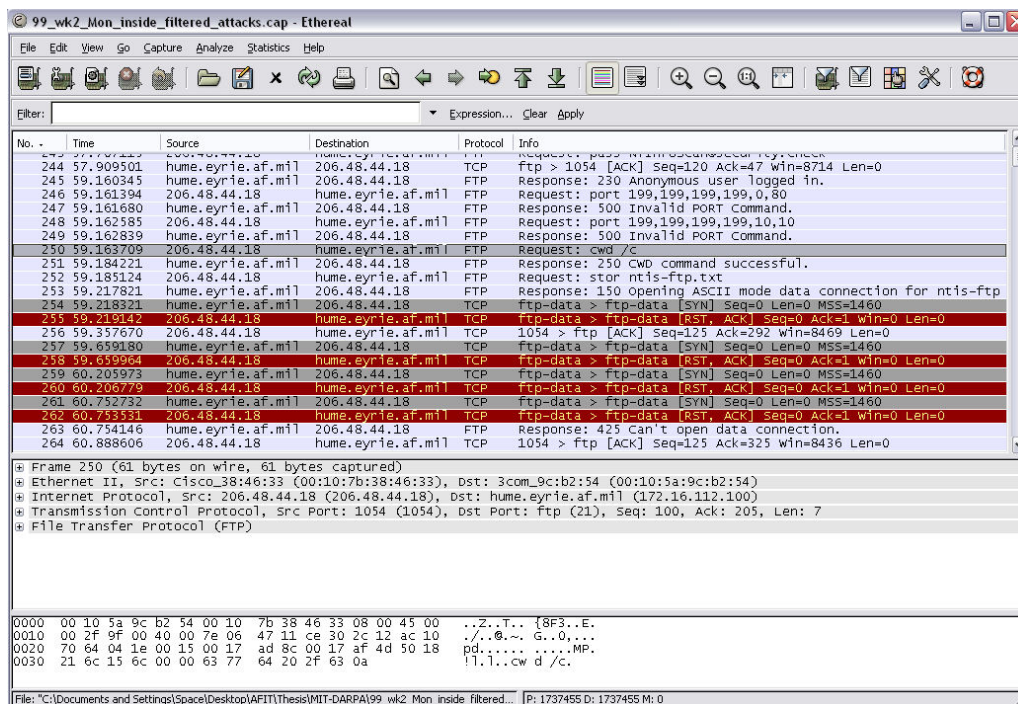


Figure 56: Example of ACO given a preponderance of food at the bottom trail

## Appendix B: MIT-DARPA 1999 Week 2 Truth Set Mapping

This appendix provides the mapping of the MIT-DARPA 1999 week-two LL truth set's high-level attack identification to what we believe to be the exact packet numbers within the second week's five data set files. In order for jREMISA to determine whether its detectors have correctly classified an incoming data set packet as *self* or *non-self*, it must be able to reference a truth table for every packet identification number. The *detections list file* provided by the LL website [MITDARPA99] provides enough high-level detail of each attack to search for it at the packet level: date, start time, destination machine, and attack technique. To discover the packet sequence and duration of an attack, we used *Ethereal* v.0.99, as shown in Figure 57.



No.	Time	Source	Destination	Protocol	Info
244	57.909501	hume.eyrie.af.mil	206.48.44.18	TCP	Request: 1034 [ACK] Seq=120 Ack=47 Win=8714 Len=0
245	59.160345	hume.eyrie.af.mil	206.48.44.18	FTP	Response: 230 Anonymous user logged in.
246	59.161394	206.48.44.18	hume.eyrie.af.mil	FTP	Request: port 199,199,199,199,0,80
247	59.161680	hume.eyrie.af.mil	206.48.44.18	FTP	Response: 500 Invalid PORT Command.
248	59.162585	206.48.44.18	hume.eyrie.af.mil	FTP	Request: port 199,199,199,199,10,10
249	59.162839	hume.eyrie.af.mil	206.48.44.18	FTP	Response: 500 Invalid PORT Command.
250	59.163709	206.48.44.18	hume.eyrie.af.mil	FTP	Request: cwd /c
251	59.184221	hume.eyrie.af.mil	206.48.44.18	FTP	Response: 250 Cwd command successful.
252	59.185124	206.48.44.18	hume.eyrie.af.mil	FTP	Request: stor ntis-ftp.txt
253	59.217821	hume.eyrie.af.mil	206.48.44.18	FTP	Response: 150 opening ASCII mode data connection for ntis-ftp
254	59.218321	hume.eyrie.af.mil	206.48.44.18	TCP	ftp-data > ftp-data [SYN] Seq=0 Len=0 MSS=1460
255	59.219142	206.48.44.18	hume.eyrie.af.mil	TCP	ftp-data > ftp-data [RST, ACK] Seq=0 Ack=1 Win=0 Len=0
256	59.357670	206.48.44.18	hume.eyrie.af.mil	TCP	1054 > ftp [ACK] Seq=125 Ack=292 Win=8469 Len=0
257	59.659180	hume.eyrie.af.mil	206.48.44.18	TCP	ftp-data > ftp-data [SYN] Seq=0 Len=0 MSS=1460
258	59.659904	206.48.44.18	hume.eyrie.af.mil	TCP	ftp-data > ftp-data [RST, ACK] Seq=0 Ack=1 Win=0 Len=0
259	60.205973	hume.eyrie.af.mil	206.48.44.18	TCP	ftp-data > ftp-data [SYN] Seq=0 Len=0 MSS=1460
260	60.206779	206.48.44.18	hume.eyrie.af.mil	TCP	ftp-data > ftp-data [RST, ACK] Seq=0 Ack=1 Win=0 Len=0
261	60.752732	hume.eyrie.af.mil	206.48.44.18	TCP	ftp-data > ftp-data [SYN] Seq=0 Len=0 MSS=1460
262	60.753531	206.48.44.18	hume.eyrie.af.mil	TCP	ftp-data > ftp-data [RST, ACK] Seq=0 Ack=1 Win=0 Len=0
263	60.754146	hume.eyrie.af.mil	206.48.44.18	FTP	Response: 425 Can't open data connection.
264	60.888606	206.48.44.18	hume.eyrie.af.mil	TCP	1054 > ftp [ACK] Seq=125 Ack=325 Win=8436 Len=0

# Frame 250 (61 bytes on wire, 61 bytes captured)  
# Ethernet II, Src: Cisco:38:46:33 (00:10:7b:38:46:33), Dst: 3com:9c:b2:54 (00:10:5a:9c:b2:54)  
# Internet Protocol, Src: 206.48.44.18 (206.48.44.18), Dst: hume.eyrie.af.mil (172.16.112.100)  
# Transmission Control Protocol, Src Port: 1054 (1054), Dst Port: ftp (21), Seq: 100, Ack: 205, Len: 7  
# File Transfer Protocol (FTP)

0000 00 10 5a 9c b2 54 00 10 7b 38 46 33 08 00 45 00 ..Z..T.. {8F3..E.  
0010 00 2f 9f 00 40 00 7e 06 47 11 ce 30 2c 12 ac 10 ..../.0.. G..0,...  
0020 70 64 04 1e 00 15 00 17 ad 8c 00 17 af 4d 50 18 pd.....MP.  
0030 21 6c 15 6c 00 00 63 77 64 20 2f 63 0a !1..!..cw d /c.

Figure 57: Ethereal analysis of the 1999 week-two Monday clean insider data set

The methodology of combining the MIT-DARPA 1999 week-two insider data set with LL’s truth set to extract an attack sequence of packets was as follows:

1. acquire and open one day’s data set in Ethereal<sup>17</sup>;
2. apply the Ethereal filter, “ip.proto == 1 || ip.proto == 17 || ip.proto == 6” (see Section 4.3.1);
3. save as “<day>\_filtered.cap”;
4. map LL’s “start time” in seconds to the Ethereal “seconds” column to verify destination IP (victim) and payload match LL truth set (where a day’s live play ranges an average of 22 hours, beginning at 0800, per LL);
5. further filter by source IP (attacker), destination IP (victim) and IP protocol (as above) to bound the attack to discover start and end packet number and time duration;
6. use jREMISA to extract each packet number into a XML file, titled by the LL-designated attack number, giving it the same filter parameters (because this author is not manually typing in 10,000 packet ID numbers).

Upon doing this for as many of the packet header-focused attacks as possible (e.g., “portsweep”), jREMISA loads the appropriate XML files into Java *TreeMaps* that perform  $O(\log n)$  search time for determining the truth of each Ab’s declaration.

---

<sup>17</sup> In May, 2006, the Ethereal project changed ownership to the open-source project, *Wireshark* (<http://www.wireshark.org>). Wireshark was unable to load data sets of our abnormally large size, hence our research stayed with the last stable release of *Ethereal*, v0.99.

Our declarations and commentary of the attack events that follow is based on empirical interpretation is what we believe to constitute the range of the attack from start to end and may not be 100% accurate. We are fairly sure of a majority accuracy, as this Cisco-Certified Network Associate (CCNA)-certified author has a decade of training and experience in network packet analysis and executing various USAF-sanctioned vulnerability scans and network-based exploits (as a former 92d Aggressor<sup>18</sup> Blue and Red Team Chief). The following tables provide the low-level mapping from the LL *detections list file* of 16 successfully extracted, context-based attacks.

**MONDAY, 03/08/99**  
**1,753,377 packets total**  
**1,737,455 feasible (TCP/UDP/ICMP = 99.09%)**

LL Truth Set Website				
ID	Date	Start	Destination	Name
2	03/08/1999	08:50:15	zeno.eyrie.af.mil	pod
	Ethereal-Mapped Interpretation			
	From-To	Duration (s)	Packets	Protocol
	206.229.221.82 > zeno.eyrie.af.mil	3011.585624 – 3011.882456	104504 – 104745 (241 consecutive packets)	ICMP (fragmented)

#### COMMENTS

1. Ethereal filter: ip.src\_host matches "206.229.221.82" && ip.dst\_host matches "zeno.eyrie.af.mil" && ip.proto == 1 (ICMP).

---

<sup>18</sup> 92d Information Warfare Aggressor Squadron, Air Force Information Warfare Center, Lackland AFB, Texas.

LL Truth Set Website				
ID	Date	Start	Destination	Name
5	03/08/1999	15:57:15	pascal.eyrie.af.mil (172.16.112.50)	land
	Ethereal-Mapped Interpretation			
	From-To	Duration (s)	Packets	Protocol
	pascal to pascal	28626.76379	Packet #1412753	TCP

#### COMMENTS

1. crafted DoS packet to make victim's address source, as well;
2. LL claims this is a UDP packet but Ethereal reports protocol as TCP.

LL Truth Set Website				
ID	Date	Start	Destination	Name
7	03/08/1999	19:09:17	pascal.eyrie.af.mil	ps attack
	Ethereal-Mapped Interpretation			
	From-To	Duration (s)	Packets	Protocol
	mars.avocado.net > pascal.eyrie.af.mil	40146.26430 – 40158.01769	695119- 695122,695124,695125, 695132, 695133	TCP-FTP

#### COMMENTS

1. pascal (victim) FTP-requests "psexp.sh.uu" then FINs the connection.

**TUESDAY, 03/09/99**

**1,585,120 packets total**

**1,571,748 feasible (TCP/UDP/ICMP = 99.15%)**

LL Truth Set Website				
ID	Date	Start	Destination	Name
8	03/09/1999	08:44:17	marx.eyrie.af.mil (153.37.134.17)	portsweep
	Ethereal-Mapped Interpretation			
	From-To	Duration (s)	Packets	Protocol
	153.37.134.17 > www.eyrie.af.mil	2653.473586 – 4269.591726	49201 – 97318 (1030 non-consecutive packets)	TCP – FIN flag

#### COMMENTS

1. LL truth set says DST\_IP = "marx.eyrie.af.mil" but Ethereal reports the FIN flag flood from marx, attacking www.eyrie.af.mil;
2. destination ports 1-1000 swept;
3. Ethereal filter: ip.src\_host matches "153.37.134.17" && ip.dst\_host matches "www.eyrie.af.mil" && ip.proto == 6 (TCP).



LL Truth Set Website				
ID	Date	Start	Destination	Name
10	03/09/1999	10:06:43	marx.eyrie.af.mil (153.37.134.17)	back
	Ethereal-Mapped Interpretation			
	From-To	Duration (s)	Packets	Protocol
	172.16.118.70 > www.eyrie.af.mil	7649.806550 – 7770.168467	185125 – 187986 (522 non-consecutive packets)	TCP-HTTP

#### COMMENTS

1. Ethereal filter: ip.src\_host matches "172.16.118.70" && ip.dst\_host matches "www.eyrie.af.mil" && ip.proto == 6 (TCP);
2. packet match entails many connections with backslash storms;
3. while LL labels this attack against marx, Ethereal reported it to be against www.eyrie.af.mil.

**WEDNESDAY, 03/10/99**

**1,011,149 packets total**

**995,235 feasible (TCP/UDP/ICMP = 98.43%)**

LL Truth Set Website				
ID	Date	Start	Destination	Name
17	03/10/1999	12:02:13	marx.eyrie.af.mil (153.37.134.17)	satan
	Ethereal-Mapped Interpretation			
	From-To	Duration (s)	Packets	Protocol
	204.97.153.43 > www.eyrie.af.mil	14537.628371 – 14672.655720	382801 – 410611 (10504 non-consecutive packets)	TCP – SYN flag

#### COMMENTS

1. Ethereal filter: ip.src\_host matches "204.97.153.43" && ip.dst\_host matches "www.eyrie.af.mil" && ip.proto == 6 (TCP);
2. looks like a SYN-based port sweep; Ethereal reports victim as "www.eyrie.af.mil" vs. LL's "marx.eyrie.af.mil";
3. SRC\_PORT +1, each time.

LL Truth Set Website				
ID	Date	Start	Destination	Name
18	03/10/1999	13:44:18	pascal.eyrie.af.mil	mailbomb
	Ethereal-Mapped Interpretation			
	From-To	Duration (s)	Packets	Protocol
	208.254.251.132 > pascal	20650.472698 – 21246.253089	555295-597287 (5004 non-consecutive packets)	TCP-SMTP

#### COMMENTS

1. Ethereal filter: ip.src\_host matches "208.254.251.132" && ip.dst\_host matches "pascal.eyrie.af.mil" && ip.proto == 6 (TCP);
2. 208.254.251.132 logs in asdfg@hotlips.com, sends a large-body email to one user @pascal.eyrie.af.mil and logs out. This occurs 500 times.

LL Truth Set Website				
ID	Date	Start	Destination	Name
22	03/10/1999	23:56:14	hume.eyrie.af.mil	crashiis
	Ethereal-Mapped Interpretation			
	From-To	Duration (s)	Packets	Protocol
	205.180.112.36 > hume	57359.536276 – 57366.408634	981138, 981140,981141,981145	TCP-HTTP

#### COMMENTS

1. as the attack describes—a single malformed HTTP packet (#981141) is sent to hume (we include the others for connection setup and teardown from attacker).

THURSDAY, 03/11/99

1,563,069 packets total

1,547,709 feasible (TCP/UDP/ICMP = 99.02%)

LL Truth Set Website				
ID	Date	Start	Destination	Name
23	03/11/1999	08:04:17	hume.eyrie.af.mil	crashiis
	Ethereal-Mapped Interpretation			
	From-To	Duration (s)	Packets	Protocol
	linux2.eyrie.af.mil > hume	240.816330 – 247.625176	3450,3452,3453,3458	TCP-HTTP

#### COMMENTS

1. as the attack describes—a single malformed HTTP packet (#3453) is sent to hume (I include the others for connection setup and teardown from attacker for patternizing attacker's source location).

LL Truth Set Website				
ID	Date	Start	Destination	Name
25	03/11/1999	10:50:11	marx.eyrie.af.mil (153.37.134.17)	satan
	Ethereal-Mapped Interpretation			
	From-To	Duration (s)	Packets	Protocol
	linux9.eyrie.af.mil > www.eyrie.af.mil	5589.224812 – 5591.868966	136873 – 157280 (10056 non-consecutive pkts)	TCP

#### COMMENTS

1. Ethereal filter: ip.src\_host matches "linux9.eyrie.af.mil" && ip.dst\_host matches "www.eyrie.af.mil" && ip.proto == 6 (TCP);
2. while labeled "satan," the pattern is "portsweep" near this Start time;
3. ethereal reports destination as "www.eyrie.af.mil", not marx;
4. SRC\_PORT increments src port +1, each time, for DST\_PORTS 1-9999.

LL Truth Set Website				
ID	Date	Start	Destination	Name
26	03/11/1999	11:04:16	pigeon.eyrie.af.mil	neptune
	Ethereal-Mapped Interpretation			
	From-To	Duration (s)	Packets	Protocol
	209.117.157.183 > pigeon	11030.776696 – 11235.663506	381781-412373 (10401 non-consecutive packets)	TCP – SYN flag

#### COMMENTS

1. Ethereal filter: ip.src\_host matches "209.117.157.183" && ip.dst\_host matches "pigeon.eyrie.af.mil" && ip.proto == 6 (TCP);
2. SYN flood: 10 packets each for DST\_PORT 1 through 1024.

LL Truth Set Website				
ID	Date	Start	Destination	Name
29	03/11/1999	15:47:15	pascal.eyrie.af.mil	land
	Ethereal-Mapped Interpretation			
	From-To	Duration (s)	Packets	Protocol
	pascal > pascal	28006.155539	Packet #1121478	TCP-SMTP

#### COMMENTS

1. as stated in the attack description, a single TCP SYN flag packet was sent where both SRC\_PORT and DST\_PORT = 25 (SMTP).

FRIDAY, 03/12/99  
1,362,422 packets total  
1,347,393 feasible (TCP/UDP/ICMP = 98.90%)

LL Truth Set Website				
ID	Date	Start	Destination	Name
35	03/12/1999	09:18:15	duck.eyrie.af.mil	pod
	Ethereal-Mapped Interpretation			
	From-To	Duration (s)	Packets	Protocol
	dialup77.glink.net.hk > duck.eyrie.af.mil	4690.999774 – 4691.534620	90303-90737 (435 consecutive packets)	ICMP (fragmented)

#### COMMENTS

1. as the attack describes—a series of contiguous, fragmented ICMP packets;
2. Ethereal filter: ip.src\_host matches "dialup77.glink.net.hk" && ip.dst\_host matches "duck.eyrie.af.mil" && ip.proto == 1.

LL Truth Set Website				
ID	Date	Start	Destination	Name
36	03/12/1999	11:20:15	marx.eyrie.af.mil (153.37.134.17)	neptune
	Ethereal-Mapped Interpretation			
	From-To	Duration (s)	Packets	Protocol
	204.97.153.43 > www.eyrie.af.mil	12010.065037 – 12214.803999	314201-342487 (10381 non-consecutive packets)	TCP – SYN flag

#### COMMENTS

1. as the attack describes, this is a SYN flood DoS, however, at this start time, the victim is www.eyrie.af.mil, not marx;
2. Ethereal filter: ip.src\_host matches "204.97.153.43" && ip.dst\_host matches "www.eyrie.af.mil" && ip.proto == 6.

LL Truth Set Website				
ID	Date	Start	Destination	Name
37	03/12/1999	12:40:12	hume.eyrie.af.mil	crashiis
	Ethereal-Mapped Interpretation			
	From-To	Duration (s)	Packets	Protocol
	alpha.apple.edu > hume	16808.70902 – 16815.70694	536909,536911, 536912,536949	TCP-HTTP

#### COMMENTS

1. packet #536912 is the malformed HTTP packet; the rest is set-up and teardown records by sender, only.

LL Truth Set Website				
ID	Date	Start	Destination	Name
42	03/12/1999	17:13:10	pascal.eyrie.af.mil	portsweep
	Ethereal-Mapped Interpretation			
	From-To	Duration (s)	Packets	Protocol
	209.167.99.71 > pascal.eyrie.af.mil	33181.082026 – 33904.161875	#1171914-1208354 (5058 non-consecutive packets)	TCP – SYN flag

#### COMMENTS

1. Ethereal filter: ip.src\_host matches "209.167.99.71" && ip.dst\_host matches "pascal.eyrie.af.mil" && ip.proto == 6.

## Appendix C: KDD Cup 99 Data Set

This appendix further explains the technical details and requirements of facilitating the KDD Cup 99 data set into jREMISA. The 1999 KDD Cup data set, used for The Third International KDD Mining Tools Competition, was held in conjunction with KDD-99 Fifth International Conference on Knowledge Discovery and Data Mining. Built upon the 1998 MIT-DARPA data sets [KDD99, Mahoney03], the competition task was to build a network intrusion detector—a predictive model capable of distinguishing between “bad” connections, called intrusions or attacks, and “good” normal connections. This database contains a standard set of data to be audited, including a wide variety of intrusions simulated in a military network environment. We desire to evaluate our algorithm against this data set, as well. However, due to lack of some basic data structure information, we were unable to.

While the MIT-DARPA data sets are binary network traffic files, each KDD Cup 99 connection record (clear-text line) is a 42-dimension clear-text array of subjective parameters based on basic features of a TCP connection and content features within a connection and traffic features within the network. The first 41 dimensions are the record composition, with the last dimension declaring whether it is a clean or attack record. While we possessed the data set and truth set, we were unable to acquire each dimension’s upper and lower bounds and discrete value definitions (i.e., some undefined values for Gene 4 include, “SF”, “S1”, “REJ”, etc.). Regardless, jREMISA includes some coded methods that prepare reading in of a KDD Cup 99 data set and selection of which fields (genes) of the record should be evaluated. Table 9 depicts our chromosomal

representation of a KDD Cup 99 connection record. Because we do not know the *Value Type*'s boundaries, we could only guess the bit lengths in the last two columns of Table 9 that would compose the Ag chromosome.

Dim (gene)	Field	Purpose	Value Type	Start Loc <sup>19</sup>	Gene Bits <sup>†</sup>
<i>Basic Features of Individual TCP Connections</i>					
1	duration	length (number of seconds) of the connection	continuous	0	16
2	protocol_type	type of the protocol, e.g. tcp, udp, etc.; author constraint: "TCP", "UDP", "ICMP" only	discrete	16	2
3	service	network service on the destination, e.g., http, telnet, etc.	discrete	18	6
4	flag	normal or error status of the connection	discrete	24	8
5	src_bytes	number of data bytes from source to destination	continuous	32	8
6	dst_bytes	number of data bytes from destination to source	continuous	40	8
7	land	1 if connection is from/to the same host/port; 0 otherwise	discrete	48	1
8	wrong_fragment	number of "wrong" fragments	continuous	49	6
9	urgent	number of urgent packets	continuous	55	6
<i>Content Features Within a Connection Suggested by Domain Knowledge</i>					
10	hot	number of "hot" indicators	continuous	61	8
11	num_failed_logins	number of failed login attempts	continuous	69	6
12	logged_in	1 if successfully logged in; 0 otherwise	discrete	75	1
13	num_compromised	number of "compromised" conditions	continuous	76	6
14	root_shell	1 if root shell is obtained; 0 otherwise	discrete	82	1
15	su_attempted	1 if "su root" command attempted; 0 otherwise	discrete	83	1
16	num_root	number of "root" accesses	continuous	84	6
17	num_file_creations	number of file creation operations	continuous	90	6
18	num_shells	number of shell prompts	continuous	96	8
19	num_access_files	number of operations on access control files	continuous	104	8
20	num_outbound_cmds	number of outbound commands in an ftp session	continuous	112	8
21	is_hot_login	1 if the login belongs to the	discrete	120	1

<sup>19</sup> This field added by this author, in development of the Ab and Ag data structures.

		``hot" list; 0 otherwise			
22	is_guest_login	1 if the login is a ``guest" login; 0 otherwise	discrete	121	1
<i>Traffic Features Computed Using a Two-Second Time Window</i>					
23	count	number of connections to the same host as the current connection in the past two seconds	continuous	122	16
24	srv_count	number of connections to the same service as the current connection in the past two seconds ( <i>same-host connection</i> )	continuous	138	16
25	serror_rate	% of connections that have ``SYN" errors ( <i>same-host connection</i> )	continuous	154	7
26	srv_serror_rate	% of connections that have ``SYN" errors ( <i>same-service connection</i> )	continuous	161	7
27	rerror_rate	% of connections that have ``REJ" errors ( <i>same-host connection</i> )	continuous	168	7
28	srv_rerror_rate	% of connections that have ``REJ" errors ( <i>same-service connection</i> )	continuous	175	7
29	same_srv_rate	% of connections to the same service ( <i>same-host connection</i> )	continuous	182	7
30	diff_srv_rate	% of connections to different services ( <i>same-host connection</i> )	continuous	189	7
31	srv_diff_host_rate	% of connections to different hosts ( <i>same-service connection</i> )	continuous	196	7
32	dst_host_count		continuous	203	16
33	dst_host_srv_count		continuous	219	16
34	dst_host_same_srv_rate		continuous	235	7
35	dst_host_diff_srv_rate		continuous	242	7
36	dst_host_same_src_port_rate		continuous	249	7
37	dst_host_srv_diff_host_rate		continuous	256	7
38	dst_host_serror_rate		continuous	263	7
39	dst_host_srv_serror_rate		continuous	270	7
40	dst_host_rerror_rate		continuous	277	7
41	dst_host_srv_rerror_rate		continuous	284	7
42	truth label: "normal" or <attackName>	Not part of data structure; testing purpose only		291	

Table 9: KDD Cup 99 data structure [adapted from KDD99]



## Appendix D: jREMISA User Manual and Source Code

This appendix is the usage guide for the jREMISA software. Program requirements, special instructions and explanation of the user interface are provided here. Complementary to this guide are descriptions next to each GUI input field, to minimize referencing this manual. Section D.1 is the “Quick Start Guide” for those who wish “out-of-the-box,” turn-key execution. Section D.2 provides full detail of all software functions. Section D.3 details the jREMISA Java files and depicts the high-level Unified Modeling Language (UML) class diagram. Section D.4 provides Source Lines of Code (SLOC) for any special programming. Section D.5 provides guidance on how to acquire this software package, which is comprised of two pieces:

1. jREMISA application (2.2 MB JAR file);
2. MIT-DARPA 1999 week-1 (clean) and week-two (attack) insider filtered<sup>20</sup> data sets for each day of both weeks (3.45 GB).

### D.1 Quick Start Guide

1. initialize population and perform *negative selection* (“Negative Selection” tab):
  - a. define the antibody population size for each IP protocol;
  - b. define the *affinity threshold*;
  - c. SELECT the jREMISA-filtered (MIT-DARPA) week-one clean data set  
(remember the day you chose);

---

<sup>20</sup> See Appendix B for filtration methodology.

- d. SELECT the absolute path and filename of the trained-and-immature negative-selected population;
- e. Click START.

2. MOEA:

- a. click the “Data Structure [MIT-DARPA]” tab;
  - i. click the IP fields you wish to be evaluated of each data set packet (by default, all are selected);
- b. click the “MOEA” tab;
- c. SELECT “Trained population file” as the just-saved trained population;
- d. SELECT the jREMISA-filtered (MIT-DARPA) week-two attack data set and choose the same day as the clean data set you chose, earlier;
- e. click the “truth set” radio button of the attack day you just chose;
- f. SELECT the path where all XML truth set files reside (should already be filled in);
- g. SELECT the absolute path and filename of the XML file that contains the final *Pareto Front* population;
- h. define the number of allowable false detections of each Ab before being removed from the population;
- i. define the percentage of Abs that is elitist-selected for secondary population (that represents the fittest of all Abs);
- j. network mode (OPTIONAL):
  - i. choose the “listen” and “broadcast” ports, pre-defined (where 1986 was the year the AIS concept was conceived);

- ii. broadcast message: optional; send messages to fellow jREMISA administrators;
- iii. broadcast nondoms(%): percentage of fittest Abs you want all other jREMISAs to consider incorporating into their population;
- iv. click the “Enable Ad-Hoc Networking” checkbox;
- k. click START.

## **D.2 User Manual**

This manual details compilation and execution details of jREMISA. When executed, jREMISA begins in the “Negative Selection” menu and has four other major function tabs, each described starting in Section D.2.2. Pressing “ERASE WINDOW” clears the console output *JTextArea*. Pressing “EXIT” cleanly exits the application (i.e., if you terminate without pressing “EXIT” leaves the app “hanging” in the COMMAND PROMPT; hence the “red-X” button is disabled). Online help is in the form of a terse usage statement of input type and bounds next to each user input field (*JTextField*).

### **D.2.1 Compiling and execution**

jREMISA is a self-contained JAR. It can be either executed from the command line or imported into a Java development environment, such as *Eclipse*, where the JAR file is decomposed into the jREMISA *project*. To execute the JAR from the command line, type “java -XX:+AggressiveHeap -jar jREMISA.jar”. When re-compiling, you should always specify the jREMISA *class* as the *main class*.

### D.2.2 Negative Selection menu (Figure 58)

- Purpose: enable user to generate trained-and-immature Ab detectors.
- Requirements:
  - clean *tcpdump* data set file;
  - detector output XML filename.
- Procedure:
  - define the primary population: either specify a prior trained-and-immature population for continued training or define the size of the TCP, UDP and ICMP primary populations;
  - define starting affinity threshold (Chapter 5 experiment results indicate 39% as producing highest classification effectiveness);
  - choose the data set to evaluate (“KDD Cup 99” is non-functional);
  - SELECT clean *tcpdump* data set absolute path and filename;
  - SELECT output file absolute path and XML filename;
  - click START.
    - “Training Population” sizes update with each passing generation;
    - pressing STOP before completion or allowing completion saves the population to the output filename specified;
    - sample *negative selection* output shown in Section 4.6, Figure 35.

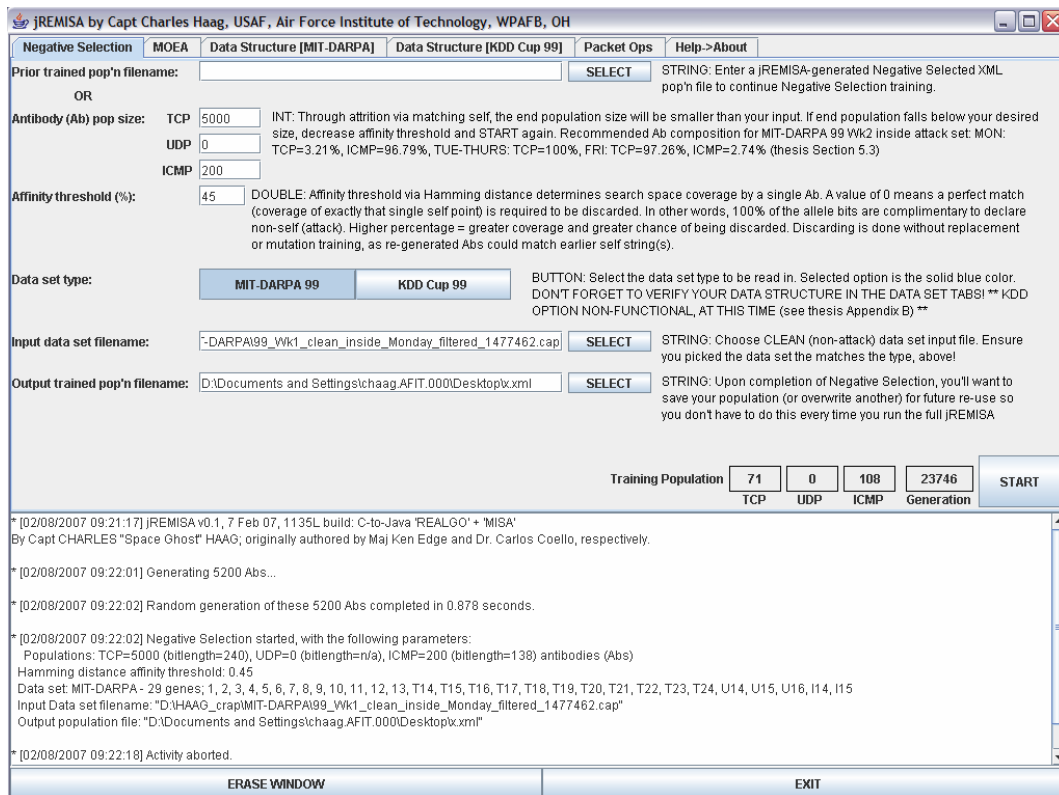


Figure 58: jREMISA *negative selection* menu

### D.2.3 MOEA Menu (Figure 59)

- Purpose: evaluate a trained (i.e., negative selected) population against an attack-filled ID data set.
- Requirements:
  - trained population XML file;
  - attack-filled data set;
  - truth set for the above attack data set;
  - post-MOEA output filename;
  - detector lifespan;
  - elitism selection percentage;

- networking mode (yes/no).
- Procedure:
  - SELECT trained population absolute path and XML filename;
  - SELECT attack-filled *tcpdump* absolute path and filename;
  - click the day of the truth set to apply to the above attack day data set;
  - SELECT the absolute path where all XML truth set files reside (double-click a filename and the field reflects absolute path, only);
  - SELECT post-MOEA output absolute path and XML filename;
  - enter detector lifespan;
  - enter elitist selection percentage;
  - click “Enable Ad-Hoc UDP Immune System Networking” if performing distributed networking:
    - distributed networking is in the form of data decomposition: equal partitions of the data set are assigned to each jREMISA;
    - Ethereal breaks the day’s data set file up into equal partitions, saved as a new *tcpdump* file, marking the start and end packet number;
    - For example, if you uniformly data-decompose Monday’s data set of 1,737,455 packets among four jREMISAs, Ethereal should save four files from the Monday data set:
      1. first file should be packet 1 – 434364;
      2. Second file should be packet 434365 – 868728;
      3. Third file should be packet 868729 – 1303092;

4. Fourth file should be packet 1303093 – 1737455,  
where each jREMISA's "Starting packet #" should be the  
starting packet number of each file it's assigned to.

○ click START.

- "Primary Population" and "Secondary Population" sizes and  
primary population effectiveness update with each passing  
generation;
- pressing STOP before completion or allowing completion saves  
the population to the output filename specified;
- sample post-MOEA output is depicted in Figure 60; it contains:
  - IP fields selected for the detector;
  - high-level effectiveness percentages with x- and y- vectors  
that can be copied-and-pasted into MATLAB variables to  
plot the attack graph, as described in Section 5.3.2;
  - For each secondary population:
    - Pareto Front x- and y-vectors that can be copied-  
and-pasted into MATLAB variables to plot the  
Pareto Front, as described in Section 5.3.2;
    - Ab DNA chromosome composition, for future  
jREMISA input.

jREMISA by Capt Charles Haag, USAF, Air Force Institute of Technology, WPAFB, OH

Negative Selection | **MOEA** | Data Structure [MIT-DARPA 99] | Data Structure [KDD Cup 99] | Packet Ops | Help->About

Trained population file: C:\TEMP\Fri42.xml SELECT STRING: trained negative selection population

Attack data set filename: FITThesis\MIT-DARPA\99\_wk2\_Fri\_inside\_filtered\_attacks.cap SELECT STRING: specify filename containing non-selfs and its truth day. If you break up the dataset among PCs, you must maintain the absolute packet#. E.g., if jREMISA\_1 does 1-5000 and jREMISA\_2 does 5001-10000, jREMISA\_2 must begin at 5001

Starting packet #: 1 SELECT

Truth XML path: file:\H:\jREMISA\jar\thesis\jREMISA\persistence\ SELECT STRING: Path to the truth set XML files that come with this program. Default path is this JAR's \persistence folder. If set to blank, entire data set is treated as self events only. Double-click any filename within the truth directory to set the truth directory.

MIT-DARPA 99 wk 2 truth day: ☒ none ☐ M ☐ T ☐ W ☐ R ☐ F SELECT

Output PF\_true filename: C:\Documents and Settings\Space\Desktop\FriPF.xml SELECT STRING: Enter base filename for PF\_true XML files. The three output files will be appended with \_TCP, \_UDP and \_ICMP, respectively

Ab lifespan:  INT: Number of false detections before deleting this Ab from the population

Selection Percentage (%): 5 DOUBLE: Percentage of best fit (most nondominated) enter ext. pop; default is 5%, as Coello recommends in MISA [CC05]

☐ Enable Ad-Hoc UDP Immune System Networking TIP: To find available ports or verify your selected ports are bound, type "netstat -an" in a COMMAND PROMPT.

Listen Port: 1986 Broadcast Port: 1987 SEND Do not use ' in messages

Broadcast message:

Broadcast Nondoms(%): 2 DOUBLE: % of nondominateds to broadcast; if 5%, enter "5"

When enabled, this AIS listens for and broadcasts its newest nondominated Abs, which go into the secondary (external) population of nondominated Abs. This decision can only be toggled when the MOEA is not running!

Self Traffic			Non-Self Traffic	
TrueNeg	FalseNeg	TruePos	FalsePos	
0.00%	0.00%	0.00%	0.00%	
Primary Population		Secondary Population		START
0	0	0	0	
TCP	UDP	ICMP	Generation	

\* [02/26/2007 13:12:41] jREMISA v0.1, 24 Feb 07, 0451L build: C-to-Java 'REALGO' + 'MISA'  
By Capt CHARLES "Space Ghost" HAAg; originally authored by Maj Ken Edge and Dr. Carlos Coello, respectively.

ERASE WINDOW EXIT

Figure 59: jREMISA MOEA menu



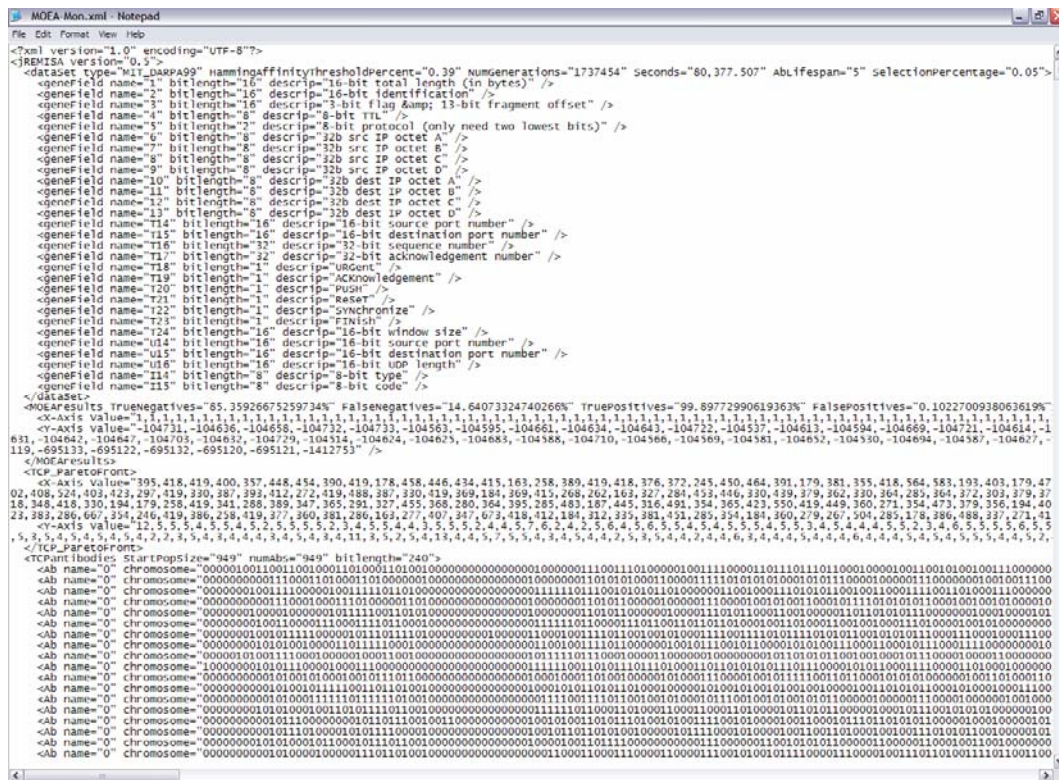


Figure 60: Example post-MOEA XML output file

#### D.2.4 “Data Structure [MIT-DARPA 99]” Menu (Figure 61)

- Purpose: define the search landscape by picking the components of the IP, TCP, UDP and ICMP packets that should be evaluated against only the same fields of the data set packets.
- Requirements: none.
- Procedure: click the fields to be evaluated; by default, all are selected.

- Requirements: none.
- Procedure: click the fields to be evaluated; by default, all are selected.

- Procedure: click the fields to be evaluated; by default, all are selected.

Figure 61: JREMISA MIT-DARPA chromosome construction menu

### D.2.5 “Data Structure [KDD Cup 99]” Menu (Figure 62)

- Purpose: define the search landscape by picking the dimensions of the 41-dimension clear-text string that should be evaluated against only the same dimensions of the data set lines.
- Requirements: none.
- Procedure: click the fields to be evaluated; by default, all are selected.
- This feature’s GUI is all that’s completed; Appendix C explains why this data set cannot be currently evaluated.

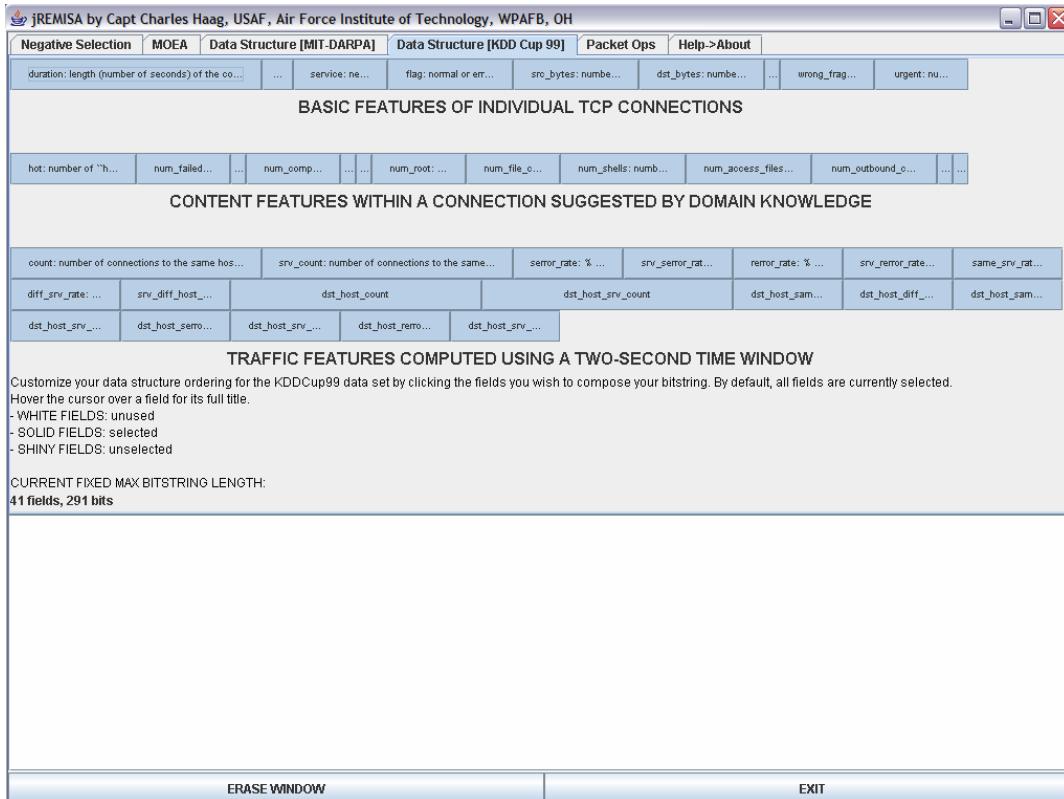


Figure 62: jREMISA KDD Cup 99 chromosome construction menu

## D.2.6 “Packet Ops” Menu (Figure 63)

- Purpose: pre-processor feature to filter and ensure entire data set can be decoded by jREMISA. If a packet is not of type TCP, UDP or ICMP, jREMISA halts, as the Java code used in decoding is only certain of when TCP, UDP and ICMP packets begin and end due to their identified sizes in their fields [Stevens94].
  - This tab was only created to prepare the data sets and is not required unless introducing new data sets.
- Requirements: data set for examination/filtration.
- Procedure: SELECT the absolute path and name of the *tcpdump* file and then select one of the three functions:

- FILTER (truth set filtration): takes a TCP/UDP/ICMP-only data set and further filters by protocol, source and destination IP and port; user additionally specifies LL attack ID, for reference, and absolute path and name of XML file to save all match packet numbers into a truth set file;
- VERIFY: attempts to read in the entire data set to ensure MOEA execution does not prematurely halt;
- INSPECT: decodes *tcpdump* file specified into clear-text output in the console window.

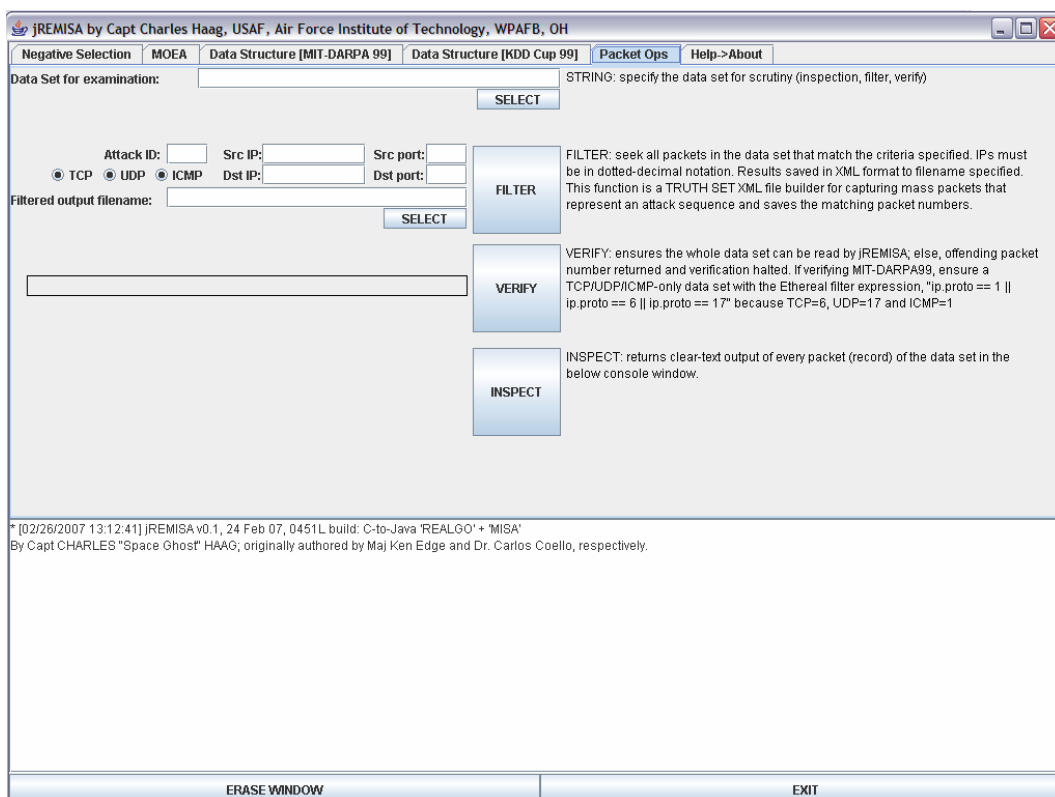


Figure 63: jREMISA *tcpdump* packet operations menu

### D.3 jREMISA file hierarchy and UML class diagram

As introduced in Section 4.1, jREMISA was built in the Eclipse IDE. It is a single project with multiple packages. Figure 64 depicts the Eclipse Package Explorer, showing the project file hierarchy. Java file variables use *Hungarian naming convention* to give developers the ability to read other people's code relatively easy, minimizing code comments [Cusumano95]. For example, in the GUI (JREMISA.java), variable names that are GUI labels are prefixed with "l\_" while GUI variable names that hold user input next to each label are prefixed with "f\_".

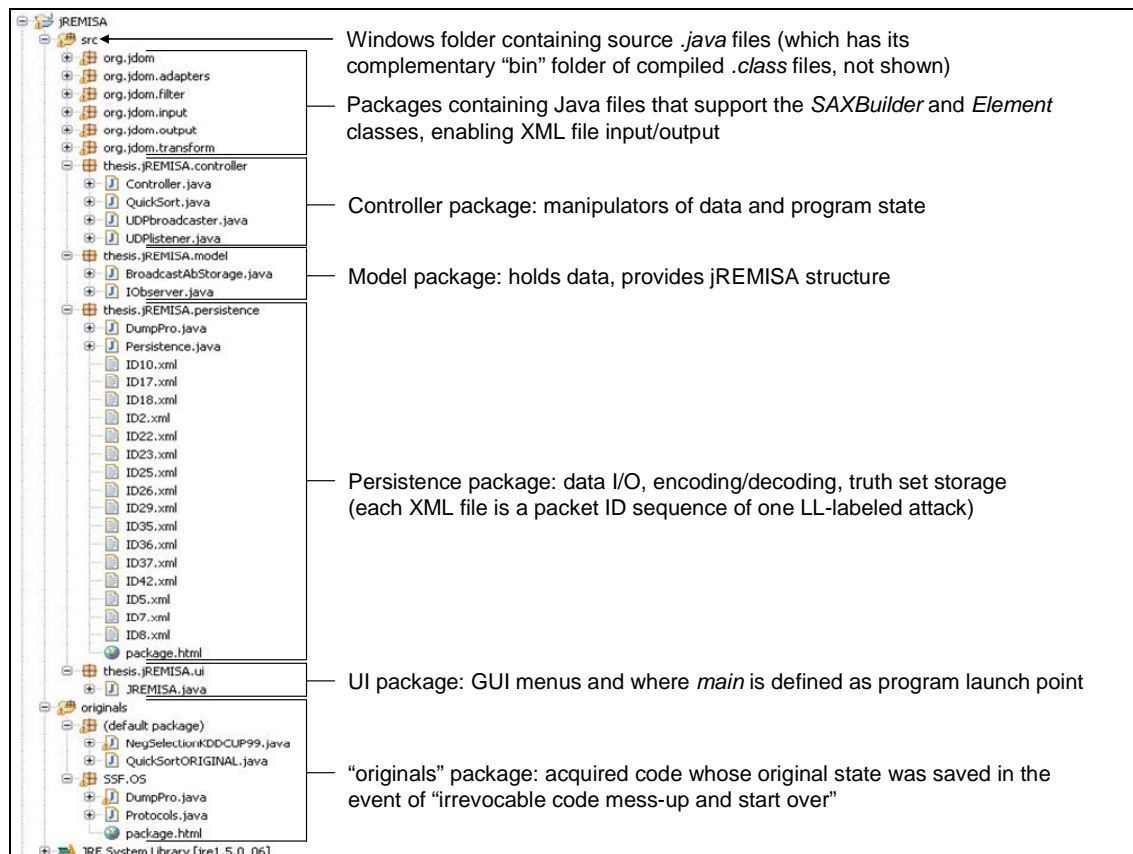


Figure 64: jREMISA file hierarchy

jREMISA functionality is made possible through the following files (Figure 64):

- *controller package*:
  - Controller.java: accepts user input from GUI, instantiating the objects, setting their parameters values and starting/stopping (if object is threaded). Enables (sets-up) and disables (tears-down) networking (sockets);
  - QuickSort.java: classic *Quicksort* algorithm, tailored to look for a particular element of the integer array to sort all Ab arrays by in ascending order in their respective *ArrayList* Ab population;
  - UDPbroadcaster.java: encodes user one-liner message or Ab into UDP packet and broadcasts to 255.255.255.255;
  - UDPlistener.java: a *Runnable* thread object that listens (*blocks*) for incoming data from the GUI-specified listen port and decodes packet. If user message, this class sends to GUI for output. Else, if Ab, sends back to *controller* for storage until end of generation, when MOEA looks in the designated *ArrayList* for any broadcast Abs.
- *model package*:
  - BroadcastAbStorage.java: *class* whose sole purpose is to maintain the *ArrayList* of captured broadcast Abs. We do this so *controller* (puts broadcast Abs in) and *dumpPro* (takes broadcast Abs out) threads can safely, independently access this container;
  - IOobserver.java: *interface* that routes all GUI-output messages from non-“UI *class*” *objects* to the GUI, for output. This implements the *Observer*

software design pattern by separating concerns between the UI and the rest of the program.

- persistence *package*:
  - DumpPro.java: this is the algorithm workhorse. Originally acquired from SSFNet (see Section 4.3.1), this *class*' original purpose was to only read in binary *tcpdump* files. To tighten code locality for faster operation, we developed all MOEA code within this *class*. As a result, this *Runnable* object is always instantiated when STARTing any functions from any of the GUI's tabbed menus. As this *class* was intended only for binary *tcpdump* files, it should not be used for data sets not using this format (i.e., KDD Cup 99);
  - Persistence.java: performs all persistent input/output. Loads and saves XML-format populations for both negative selection and MOEA operations and saves packet filter matches as an XML file containing packet identification numbers;
  - 16 "ID<#>.xml" files: all 16 attacks' extracted packet numbers (via the Filter function in the GUI) from the LL week two data sets are each saved into a XML file with the LL-labeled attack number as the filename.
- ui *package*:
  - JREMISA.java: the GUI and program execution point (*main class*).

- “originals” *package* (no Java files in this package participate in jREMISA):
  - QuickSortORIGINAL: Internet-acquired code treated as the original copy;
  - NegSelectionKDDCUP99: this class was intended to decode and parse the KDD Cup 99 data; it’s started but not finished;
  - *SSF.OS* package: Internet-acquired code treated as the original copy.

Figure 65 depicts jREMISA’s UML class diagram in the MVC architecture. For the sake of brevity and keeping the diagram to one page, attributes and methods are not included, other than the *main class* to indicate program launch point.

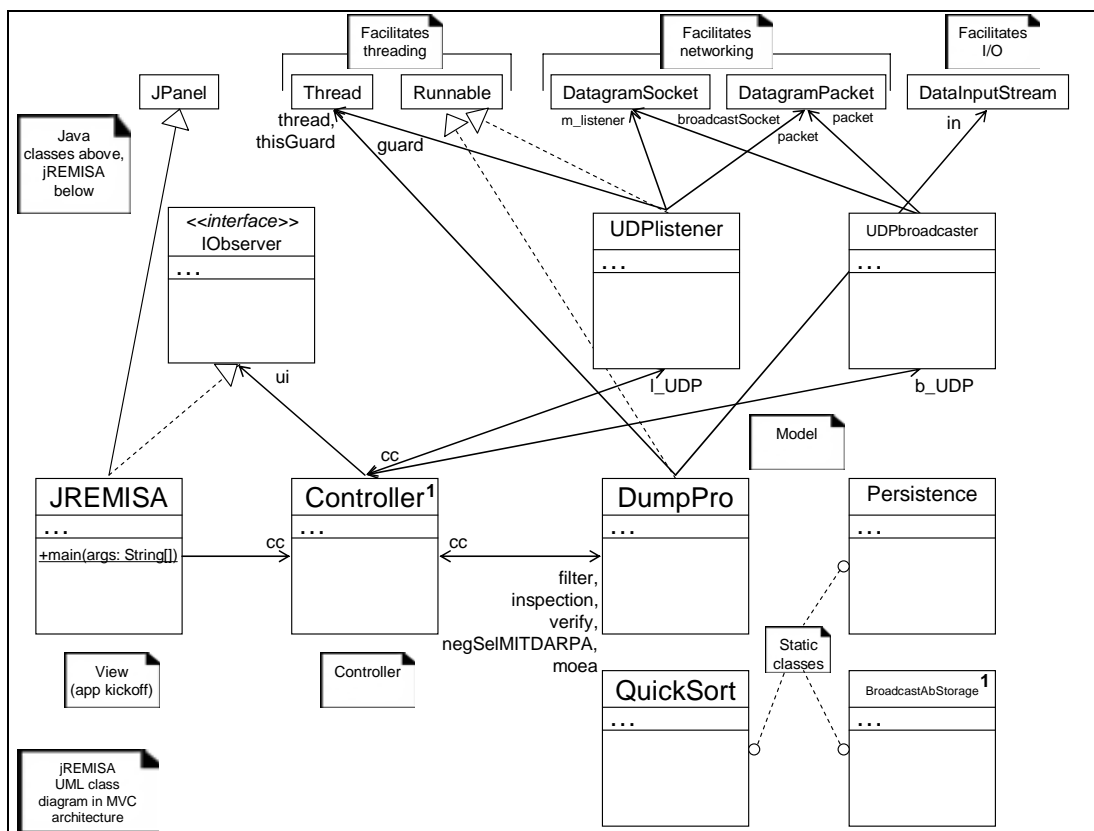


Figure 65: jREMISA UML class diagram



## D.4 Special Source Code

We inserted Windows API system calls into *misa.c* and *realgo.cpp* to acquire a nanosecond-precision timer for the purpose of comparing efficiency to its Java counterpart (see Section 4.2):

```
// The below three lines aid in acquiring system time on the
nanosecond-level by accessing the Win32 API
// WARNING: This method of time-capture is effective only on single
core CPU architectures
#pragma comment(lib, "winmm.lib")    // Additionally link this lib (same
as adding it in Config settings)
#include <windows.h>
#include <mmsystem.h>
...
// Get the high resolution counter's accuracy
QueryPerformanceFrequency(&ticksPerSecond);
QueryPerformanceCounter(&startClock);    // MARK START TIME
...
QueryPerformanceCounter(&endClock);    // MARK END TIME

printf( "elapsed: %3.6f ms\n", ((double)(endClock.QuadPart -
startClock.QuadPart) / (double)ticksPerSecond.QuadPart) * 1000);
```

## D.5 Source Code Availability

The source code for jREMISA and accompanying filtered MIT-DARPA 1999 week-one and week-two data sets are not included as part of this document. Those interested in obtaining a copy of either should direct their request to:

Dr. Gary B. Lamont

AFIT/ENG

2950 Hobson Way, Building 640

WPAFB, OH 45433-7765

Gary.Lamont@afit.edu

## Appendix E: Recommended Software Engineering Principles

This appendix is motivated by this researcher's personal experience of constantly acquiring software by others who code as if they never use it again. Such symptoms of software engineering apathy include:

1. lack of regular commenting of code;
2. no “quick start” or compile guide;
3. using special software libraries (i.e., Dynamically-Linked Libraries or DLLs) without indicating;
4. hard-wiring parameters and variables, preventing dynamic reconfiguration without having to re-compile each time;
5. not including raw output files with software when surrendered to academic institution.

Applications that are devoid of compile and execution help, usage statements and source code commenting increase the software learning curve, consequently lessening the desire to inherit such an application. The Institute of Electrical and Electronics Engineers (IEEE) Standard 610.12-1990 defines *software engineering* as, “(1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1).” This definition implies constrained coding practices in developing concisely written and understood software applications. Such practices can be applied equally by software engineers and non-software engineers, alike, and reduces the

learning curve, resulting in more time for application development. Hence, we recommend the following basic software engineering practices:

1. ensure finished application compiles on school's common lab computer—not just the developer's personal computer;
2. comment the code:
  - a. a terse paragraph at the top of the program file explaining its intent;
  - b. a comment summary of each method (function);
  - c. if employing a GUI, one-liner usage comment pop-up when the mouse or cursor hovers over an input field or usage help next to the input field;
3. include a help statement explaining both execution and compile instructions and a usage statement of all arguments and parameters;
4. include raw data files, in addition to source code, in the final software package;
5. avoid defining (“hardwiring”) values of variables in the code; allow for command line arguments or GUI fields to facilitate changing values at runtime, without recompiling.

Following these practices preserves software for future use by author and successor, in both academia and real-world applications. The longer the time passed between reusing code, the greater appreciation one has in more quickly understanding the reason and manner in which the code was written. In summary, this author's golden rule is “code it as if your work is carried on.”

## Appendix F: The Benefits of Open-Source Software

This appendix is motivated by this researcher's inability to acquire ID evaluation results, signatures and data sets from proprietary sources such as anti-virus software development companies. While the need to keep company secrets has merit, there exist ways to still work with the leading ID software developers of the day. By not facilitating an open dialogue, our few aging data sets continue to be the ID application developer's only benchmark against today's new breed of attacks. If commercial entities still refuse to communicate, then perhaps an open-source development approach needs to be taken.

To date, there have been many public debates, case studies and even an AFIT-sanctioned course this researcher completed that contrasted the value and risk of *open-source*, public domain source code versus proprietary software [Raymond00, MFH02, HS02, HSE03]. One of the shortfalls of this software's development is the inability to sample IDS manufacturers' signature generating and comparison methodologies. Therefore, we recommend jREMISA's lifecycle continue in an open-source manner for the following reasons:

1. the prospect of free and conveniently available software entices more curious people to experiment with this work;
2. multiple parties can develop it, while openly communicating ideas to each other and improving the existing code;
3. public domain source code minimizes the possibility of malicious code or exploitable vulnerabilities;

4. the upgrade cycle is tightened due to less required formality (e.g., no marketing and procedure for costly upgrading), allowing for quicker source code releases.

## **Vita**

Major (select) Charles R. Haag was commissioned in the Air Force in 1998 through the Reserve Officers Training Corps (ROTC) program at the Illinois Institute of Technology, Detachment 195, where he earned his Bachelor of Science degree in Computer Science, graduating “with honors.” From 2002-2005, Maj (sel) Haag had the distinct privilege of serving with the 92d Information Warfare Aggressor Squadron as a certified Blue and Red Team Chief and Red Team Operator Course (RTOC) cadre member, performing network and physical vulnerability assessments at Air Force installations, worldwide. Maj (sel) Haag has been awarded the Meritorious Service Medal, the Air Force Commendation Medal and the Air Force Achievement Medal. Upon graduation, Maj (sel) Haag will be assigned to the 83d Communications Squadron at Langley Air Force Base, Virginia.

<b>REPORT DOCUMENTATION PAGE</b>				Form Approved OMB No. 074-0188	
<p>The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</p> <p><b>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</b></p>					
1. REPORT DATE (DD-MM-YYYY) 22-03-2007		2. REPORT TYPE Master's Thesis		3. DATES COVERED (From – To) January 2006 – March 2007	
4. TITLE AND SUBTITLE  AN ARTIFICIAL IMMUNE SYSTEM-INSPIRED MULTIOBJECTIVE EVOLUTIONARY ALGORITHM WITH APPLICATION TO THE DETECTION OF DISTRIBUTED COMPUTER NETWORK INTRUSIONS				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)  Haag, Charles, R., Captain, USAF				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way, Building 640 WPAFB OH 45433-8865				8. PERFORMING ORGANIZATION REPORT NUMBER  AFIT/GCS/ENG/07-05	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Ms. Christine M. Nickell, (comm) 410-854-6206, (fax) 410-854-7043 Chief, Academic Outreach (I02E) ATTN: I02E, Ste 6744, 9800 Savage Rd, Ft. Meade, MD 20755-6744 c.nicke2@radium.ncsc.mil				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT  APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT <p>Today's predominantly-employed signature-based intrusion detection systems are reactive in nature and storage-limited. Their operation depends upon catching an instance of an intrusion or virus after a potentially successful attack, performing post-mortem analysis on that instance and encoding it into a signature that is stored in its anomaly database. The time required to perform these tasks provides a window of vulnerability to DoD computer systems. Further, because of the current maximum size of an Internet Protocol-based message, the database would have to be able to maintain <math>256^{65535}</math> possible signature combinations. In order to tighten this response cycle within storage constraints, this thesis presents an <i>Artificial Immune System-inspired Multiobjective Evolutionary Algorithm</i> intended to measure the vector of tradeoff solutions among detectors with regard to two independent objectives: best classification fitness and optimal hypervolume size. Modeled in the spirit of the human biological immune system and intended to augment DoD network defense systems, our algorithm generates network traffic detectors that are dispersed throughout the network. These detectors promiscuously monitor network traffic for exact and variant abnormal system events, based on only the detector's own data structure and the ID domain truth set, and respond heuristically.</p> <p>The application domain employed for testing was the MIT-DARPA 1999 intrusion detection data set, composed of 7.2 million packets of notional Air Force Base network traffic. Results show our proof-of-concept algorithm correctly classifies at best 86.48% of the normal and 99.9% of the abnormal events, attributed to a detector affinity threshold typically between 39-44%. Further, four of the 16 intrusion sequences were classified with a 0% false positive rate.</p>					
15. SUBJECT TERMS intrusion detection, computer networks, stochastic search, computer security, evolutionary computation, artificial immune system, multiobjective evolutionary algorithm					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Gary B. Lamont, PhD, AFIT/ENG
U	U	U	UU	224	19b. TELEPHONE NUMBER (Include area code) (937) 785-3636, x4718 (gary.lamont@afit.edu)

